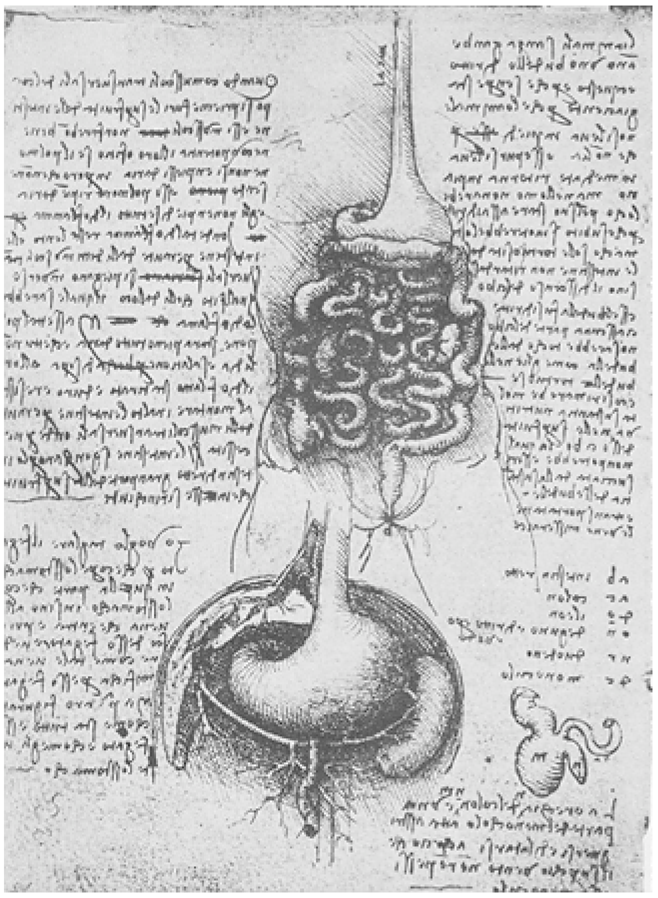
The Guts of LON-CAPA

Workshop

Michigan State University
June 10th-12th, 2002



This project is funded in part by the
National Science Foundation under ITR 0085921,
with additional support by the
Andrew W. Mellon and Alfred P. Sloan foundations.

http://www.lon-capa.org/
(517) 432-5468

Guy Albertelli, albertel@msu.edu
Gerd Kortemeyer, korte@lon-capa.org
Scott Harrison, freeware volunteer, sharrison@sourceforge.net

Project Manager: Helen Keefe, helen@loncapa.org
LON-CAPA Coordinator: Felicia Berryman, felicia@lon-capa.org

Installation site: http://install.lon-capa.org/
Mailing lists: http://mail.lon-capa.org/
User help: http://help.lon-capa.org/
Bugs and enhancements: http://bugs.lon-capa.org/

DAY 1 ..... **6**

SESSION ONE: INTRO/DEMO, LONC/D, REPLICATION AND LOAD BALANCING (GERD) 6
Overview *7*
Example of Transactions *7*
lonc/lond/lonnet *7*
Scalability and Performance Analysis *10*
Dynamic Resource Replication *12*
SESSION TWO: APACHE HANDLERS (LONCAPA_APACHE.CONF), AUTHENTICATION AND ACCESS CONTROL, SESSION ENVIRONMENT (MATTHEW) 16
Server Content Resource Areas *16*
Apache Request Cycle and Handlers *16*
Handler Definition *17*
Authentication Overview *18*
Authentication Mechanisms, User Data, Passwords *19*
Environment *19*
Handler Reference: LON-CAPA and the 77 Web Perl Modules *22*
SESSION THREE: ROLES AND LONNET/LONCOMMON USEFUL FUNCTIONS (GUY) 29
NAME *29*
SYNOPSIS *29*
INTRODUCTION *29*
RETURN MESSAGES *29*
Session Environment Functions *30*
User Information *30*
User Roles *30*
User Modification *30*
Course Infomation *31*
Course Modification *31*
Resource Subroutines *31*
Resource Information *31*
Storing/Retrieving Data *32*
Network Status Functions *33*
Apache Request *33*
Data to String to Data *33*
Logging Routines *34*
General File Helper Routines *34*
HTTP Helper Routines *34*
Private: Underlying communication routines (Shouldn't call) *34*
Private: Resource Access Logging *35*
Private: Other *35*
Coding Guidelines *35*
SESSION FOUR: HOW TO INSTALL, PROPER DEBUG CYCLE FOR HANDLERS, CODING/DOCUMENTATION GUIDELINES (GUY) 36
lpml.dtd *36*
piml.dtd *38*
xfml.dtd *40*
SESSION FIVE: WORKTIME (MAKE NEW HANDLER, DEBUGGING/TESTING) (GUY) 41

DAY 2 ..... **42**

SESSION ONE: ROLES, DATA STORAGE, PARAMETERS (GERD) 42
Domains *42*
Userdata *42*
Courses *43*
Roles *43*
Custom Roles *44*

Choose a Role, Role Privileges *44*
Role Initialization *46*
SESSION TWO: SPREADSHEET AND MESSAGING (MATTHEW) 48
Spreadsheets *48*
Spreadsheet Structure and Hierarchy *48*
Export Rows *49*
Assessment Spreadsheet *49*
Student Spreadsheet *50*
Course Spreadsheet *50*
Spreadsheet definition *51*
Template Row *51*
What goes in a cell *52*
Parameter Access *52*
Addressing other cells *52*
The Safe Environment *53*
Change is coming *53*
Discussion *54*
Messaging *55*
Resource Feedback *56*
SESSION THREE: PUBLICATION, CONTENT MAPS, COURSE MAPS (GERD) 56
Publication of a Resource *56*
Content Re-usage and Granularity *57*
Maps *58*
Curriculum Adaptivity *58*
Resource Assembly Tool *59*
Map Representation and Storage Format *61*
Example of Nested Maps *62*
Initialization of a Course for a Learner *65*
Evaluation of the Map Structure for a Course *66*
Paths and Path Conditions *70*
Multivalued Boolean Evaluation of Link Priorities *73*
SESSION FOUR: XML HANDLER (SIMPLE TAGS, GLOBALS, MULTIPLE TARGETS, STYLE FILES) (GUY) 74
XML Files *74*
Style Files *76*
HTML::LCParser - Alternative HTML::Parser interface *77*
SESSION FIVE: WORKTIME (GUY) 80

DAY 3 .... **81**

SESSION ONE: PROBLEM ENGINE/SPECIAL TARGETS (GRADE, EDIT, PRINT) (GUY) 81
Tags *81*
<script> Functions *86*
Detailed descriptions of each function and comparison with CAPA *87*
<script> Variables *93*
Form internals *94*
Viewing a problem (either CSTR or RES space) *94*
Editing a problem (Edit mode) *94*
Symbs *95*
Store / Restore *95*
Mandatory Homework Data *97*
Sample Problems *98*
SESSION TWO: STATISTICS/CHART (BEHROUZ) 101
lonchart.pm *102*
lonstatistics.pm *102*
Problem Analysis *106*
Student Assessment *110*
Future work, using activity.log to classify the students *111*

SESSION THREE: LONSQL (GERD) 113
Purpose within LON-CAPA *113*
SESSION FOUR: CVS/BUGZILLA INTRO (GUY) 114
SESSION FIVE: OPEN WORKTIME 114

DIFFERENT TYPES OF LON-CAPA SERVERS 115

INTRODUCTION 115
LIBRARY SERVERS 115
ACCESS SERVERS 115

HARDWARE SPECIFICATION FOR LON-CAPA SERVERS 115

GENERAL INFORMATION ON LON-CAPA SERVER HARDWARE REQUIREMENTS 115
ADDITIONAL INFORMATION ON LON-CAPA SERVER HARDWARE REQUIREMENTS 116

LON-CAPA ON THE LINUX FILESYSTEM 116

1. INTRODUCTION 116
2. THE LON-CAPA SOURCE FILES 117
3. ACTIVE FILESYSTEM DIRECTORIES OF IMPORTANCE 118
4. THE INSTALLATION MECHANISM 119
4.1 Standard procedure *119*
4.2 XML specification *120*
4.3 Makefile *120*
5. AUTOMATED TESTING 121
5.1 Standard Procedure for Testing *121*
5.2 Testing the System Services (MySQL, perl libraries, etc) *122*
5.3 Testing the LON-CAPA Web Interface *122*
5.4 Testing the Installation Mechanism *122*
6. SOFTWARE PACKAGES 123
7. SECURITY ON THE FILESYSTEM 123
Make a LON-CAPA system work with shadow passwords *125*

CONFIGURATION 126

CONFIGURING THE APACHE WEB SERVER 126
CONFIGURING THE LON-CAPA NETWORK 127
CONFIGURING MYSQL 128

FAST INSTALLATION: SETTING UP A RED HAT 7.3 LON-CAPA SERVER 129

BEFORE YOU BEGIN 129
INSTALLATION OVERVIEW 129
OBTAIN THE RED HAT 7.3 INSTALLATION CDS 129
DETERMINE NETWORK SETTINGS FOR YOUR SITE 129
INSTALL RED HAT 7.3 129
DETERMINE LON-CAPA SETTINGS 130
INSTALL LON-CAPA 130
CONFIGURE LON-CAPA 131
PICKING A HOSTS.TAB FILE 131
CREATING A DOMAIN COORDINATOR 131
START/RESTART SERVICES 131
MAKE SURE THAT SERVICES AUTOSTART ON BOOT 131
LOG IN TO YOUR LON-CAPA MACHINE 132

GLOSSARY 133

Day 1

Session One: Intro/Demo, lonc/d, Replication and Load Balancing (Gerd)

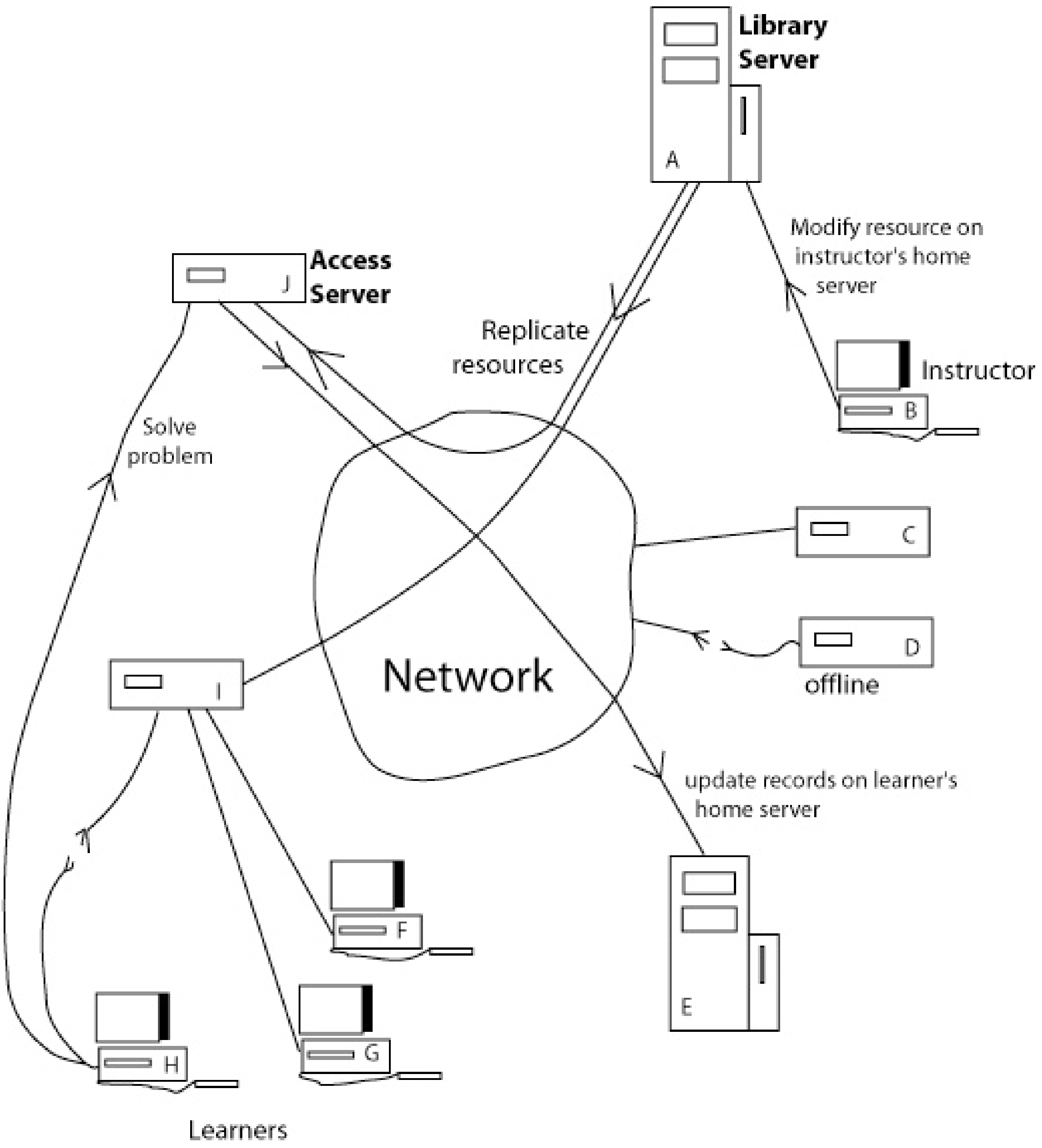


Fig. 1.1.1 – Overview of Network

Overview

Physically, the Network consists of relatively inexpensive upper-PC-class server machines which are linked through the commodity internet in a load-balancing, dynamically content-replicating and failover-secure way. **Fig. 1.1.1** shows an overview of this network.

All machines in the Network are connected with each other through two-way persistent TCP/IP connections. Clients (**B**, **F**, **G** and **H** in **Fig. 1.1.1**) connect to the servers via standard HTTP. There are two classes of servers, Library Servers (**A** and **E** in **Fig. 1.1.1**) and Access Servers (**C**, **D**, **I** and **J** in **Fig. 1.1.1**). Library Servers are used to store all personal records of a set of users, and are responsible for their initial authentication when a session is opened on any server in the Network. For Authors, Library Servers also hosts their construction area and the authoritative copy of the current and previous versions of every resource that was published by that author. Library servers can be used as backups to host sessions when all access servers in the Network are overloaded. Otherwise, for learners, access servers are used to host the sessions. Library servers need to be strong on I/O, while access servers can generally be cheaper hardware. The network is designed so that the number of concurrent sessions can be increased over a wide range by simply adding additional Access Servers before having to add additional Library Servers. Preliminary tests showed that a Library Server could handle up to 10 Access Servers fully parallel.

The Network is divided into so-called domains, which are logical boundaries between participating institutions. These domains can be used to limit the flow of personal user information across the network, set access privileges and enforce royalty schemes.

Example of Transactions

Fig. 1.1.1 also depicts examples for several kinds of transactions conducted across the Network.

An instructor at client **B** modifies and publishes a resource on her Home Server **A**. Server **A** has a record of all server machines currently subscribed to this resource, and replicates it to servers **D** and **I**. However, server **D** is currently offline, so the update notification gets buffered on **A** until **D** comes online again. Servers **C** and **J** are currently not subscribed to this resource.

Learners **F** and **G** have open sessions on server **I**, and the new resource is immediately available to them.

Learner **H** tries to connect to server **I** for a new session, however, the machine is not reachable, so he connects to another Access Server **J** instead. This server currently does not have all necessary resources locally present to host learner **H**, but subscribes to them and replicates them as they are accessed by **H**.

Learner **H** solves a problem on server **J**. Library Server **E** is **H**'s Home Server, so this information gets forwarded to **E**, where the records of **H** are updated.

lonc/lond/lonnet

Fig. 1.1.2 elaborates on the details of this network infrastructure.

Fig. 1.1.2A depicts three servers (**A**, **B** and **C**, **Fig. 1.1.2A**) and a client who has a session on server **C.**

As **C** accesses different resources in the system, different handlers, which are incorporated as modules into the child processes of the web server software, process these requests.

Our current implementation uses `mod_perl` inside of the Apache web server software. As an example, server **C** currently has four active web server software child processes. The chain of handlers dealing with a certain resource is determined by both the server content resource area (see below) and the MIME type, which in turn is determined by the URL extension. For most URL structures, both an authentication handler and a content handler are registered.

Handlers use a common library `lonnet` to interact with both locally present temporary session data and data across the server network. For example, `lonnet` provides routines for finding the home server of a user, finding the server with the lowest loadavg, sending simple command-reply sequences, and sending critical messages such as a homework completion, etc. For a non-critical message, the routines reply with a simple "connection lost" if the message could not be delivered. For critical messages, `lonnet` tries to re-establish connections, re-send the command, etc. If no valid reply could be received, it answers "connection deferred" and stores the message in buffer space to be sent at a later point in time. Also, failed critical messages are logged.

The interface between `lonnet` and the Network is established by a multiplexed UNIX domain socket, denoted DS in **Fig. 1.1.2A**. The rationale behind this rather involved architecture is that httpd processes (Apache children) dynamically come and go on the timescale of minutes, based on workload and number of processed requests. Over the lifetime of an httpd child, however, it has to establish several hundred connections to several different servers in the Network.

On the other hand, establishing a TCP/IP connection is resource consuming for both ends of the line, and to optimize this connectivity between different servers, connections in the Network are designed to be persistent on the timescale of months, until either end is rebooted. This mechanism will be elaborated on below.

Establishing a connection to a UNIX domain socket is far less resource consuming than the establishing of a TCP/IP connection. `lonc` is a proxy daemon that forks off a child for every server in the Network. . Which servers are members of the Network is determined by a lookup table, which **Fig. 1.1.2B** is an example of. In order, the entries denote an internal name for the server, the domain of the server, the type of the server, the host name and the IP address.

The `lonc` parent process maintains the population and listens for signals to restart or shutdown, as well as *USR1*. Every child establishes a multiplexed UNIX domain socket for its server and opens a TCP/IP connection to the `lond` daemon (discussed below) on the remote machine, which it keeps alive. If the connection is interrupted, the child dies, whereupon the parent makes several attempts to fork another child for that server.

When starting a new child (a new connection), first an init-sequence is carried out, which includes receiving the information from the remote `lond` which is needed to establish the 128-bit encryption key – the key is different for every connection. Next, any buffered (delayed) messages for the server are sent.

In normal operation, the child listens to the UNIX socket, forwards requests to the TCP connection, gets the reply from `lond`, and sends it back to the UNIX socket. Also, `lonc` takes care to the encryption and decryption of messages.

`lonc` was build by putting a non-forking multiplexed UNIX domain socket server into a framework that forks a TCP/IP client for every remote `lond`.

`lond` is the remote end of the TCP/IP connection and acts as a remote command processor. It receives commands, executes them, and sends replies. In normal operation, a `lonc` child is constantly connected to a dedicated `lond` child on the remote server, and the same is true vice versa (two persistent connections per server combination).

`lond` listens to a TCP/IP port (denoted P in **Fig. 1.1.2A**) and forks off enough child processes to have one for each other server in the network plus two spare children. The parent process maintains the population and listens for signals to restart or shutdown. Client servers are authenticated by IP.

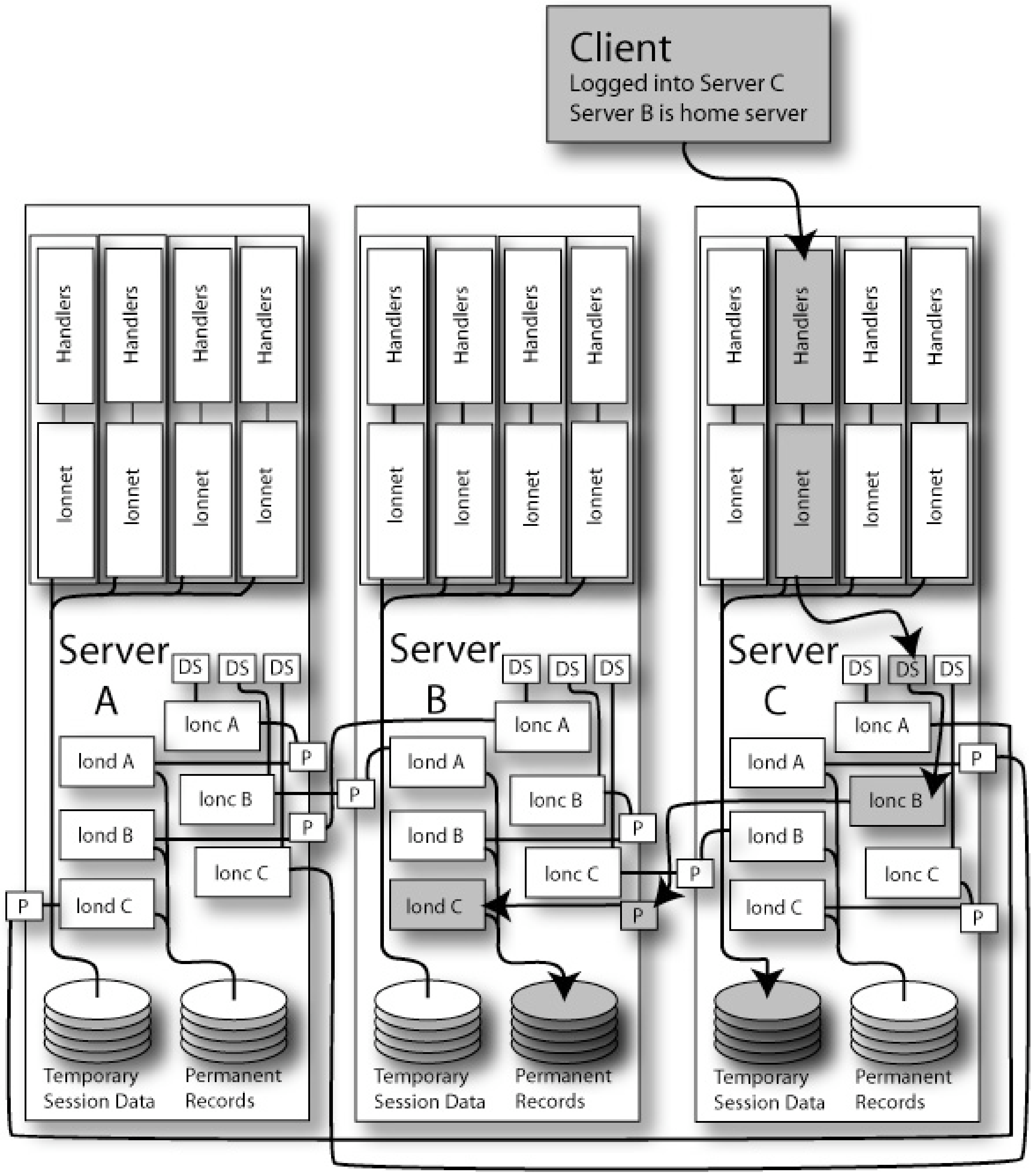


Fig. 1.1.2A – Overview of Network Communication

When a new client server comes online*,* `lond` sends a signal *USR1* to `lonc`, whereupon `lonc` tries again to reestablish all lost connections, even if it had given up on them before – a new client connecting could mean that that machine came online again after an interruption.

The gray boxes in **Fig. 1.1.2A** denote the entities involved in an example transaction of the Network. The Client is logged into server **C**, while server **B** is her Home Server. Server **C** can be an Access Server or a Library Server, while server **B** is a Library Server. She submits a solution to a homework problem, which is processed by the appropriate handler for the MIME type "problem". Through `lonnet`, the handler writes information about this transaction to the local session data. To make a permanent log entry, `lonnet` establishes a connection to the UNIX domain socket for server **B**. lonc receives this command, encrypts it, and sends it through the persistent TCP/IP connection to the TCP/IP port of the remote lond. lond decrypts the command, executes it by writing to the permanent user data files of the client, and sends back a reply regarding the success of the operation. If the operation was unsuccessful, or the connection would have broken down, lonc would write the command into a FIFO buffer stack to be sent again later. lonc now sends a reply regarding the overall success of the operation to lonnet via the UNIX domain port, which is eventually received back by the handler.

Scalability and Performance Analysis

The scalability was tested in a test bed of servers between different physical network segments, **Fig. 1.1.2B** shows the network configuration of this test.

```
msul1:msu:library:zaphod.lite.msu.edu:35.8.63.51
msua1:msu:access:agrajag.lite.msu.edu:35.8.63.68
msul2:msu:library:frootmig.lite.msu.edu:35.8.63.69
msua2:msu:access:bistromath.lite.msu.edu:35.8.63.67
hubl14:hub:library:hubs128-pc-14.cl.msu.edu:35.8.116.34
hubl15:hub:library:hubs128-pc-15.cl.msu.edu:35.8.116.35
hubl16:hub:library:hubs128-pc-16.cl.msu.edu:35.8.116.36
huba20:hub:access:hubs128-pc-20.cl.msu.edu:35.8.116.40
huba21:hub:access:hubs128-pc-21.cl.msu.edu:35.8.116.41
huba22:hub:access:hubs128-pc-22.cl.msu.edu:35.8.116.42
huba23:hub:access:hubs128-pc-23.cl.msu.edu:35.8.116.43
hubl25:other:library:hubs128-pc-25.cl.msu.edu:35.8.116.45
huba27:other:access:hubs128-pc-27.cl.msu.edu:35.8.116.47
```

Fig. 1.1.2B – Example of Hosts Lookup Table /home/httpd/lonTabs/hosts.tab

In the first test, the simple ping command was used. The ping command is used to test connections and yields the server short name as reply. In this scenario, lonc was expected to be the speed-determining step, since lond at the remote end does not need any disk access to reply. The graph **Fig. 1.1.2C** shows number of seconds till completion versus number of processes issuing 10,000 ping commands each against one Library Server (450 MHz Pentium II in this test, single IDE HD). For the solid dots, the processes were concurrently started on *the same* Access Server and the time was measured till the processes finished – all processes finished at the same time. One Access Server (233 MHz Pentium II in the test bed) can process about 150 pings per second, and as expected, the total time grows linearly with the number of pings.

The gray dots were taken with up to seven processes concurrently running on *different* machines and pinging the same server – the processes ran fully concurrent, and each process finished as if the other ones were not present (about 1000 pings per second). Execution was fully parallel.

In a second test, lond was the speed-determining step – 10,000 put commands each were issued first from up to seven concurrent processes on the same machine, and then from up to seven processes on different machines. The put command requires data to be written to the permanent record of the user on the remote server.

In particular, one "put" request meant that the process on the Access Server would connect to the UNIX domain socket dedicated to the library server, lonc would take the data from there, shuffle it through the persistent TCP connection, lond on the remote library server would take the data, write to disk (both to a dbm-file and to a flat-text transaction history file), answer "ok", lonc would take that reply and send it to the domain socket, the process would read it from there and close the domain-socket connection.

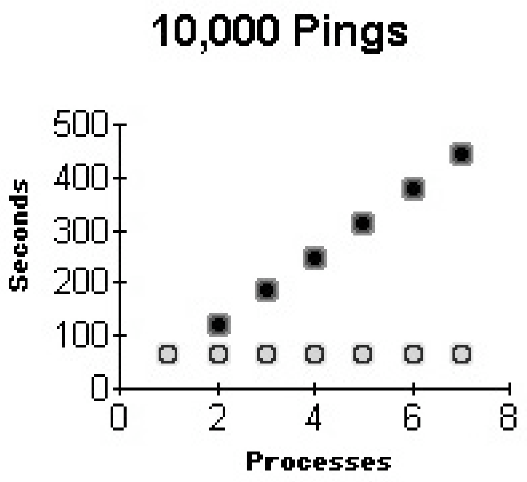


Fig. 1.1.2C – Benchmark on Parallelism of Server-Server Communication (no disk access)

The graph **Fig. 1.1.2D** shows the results. Series 1 (solid black diamond) is the result of concurrent processes on the same server – all of these are handled by the same server-dedicated `lond`-child, which lets the total amount of time grow linearly.

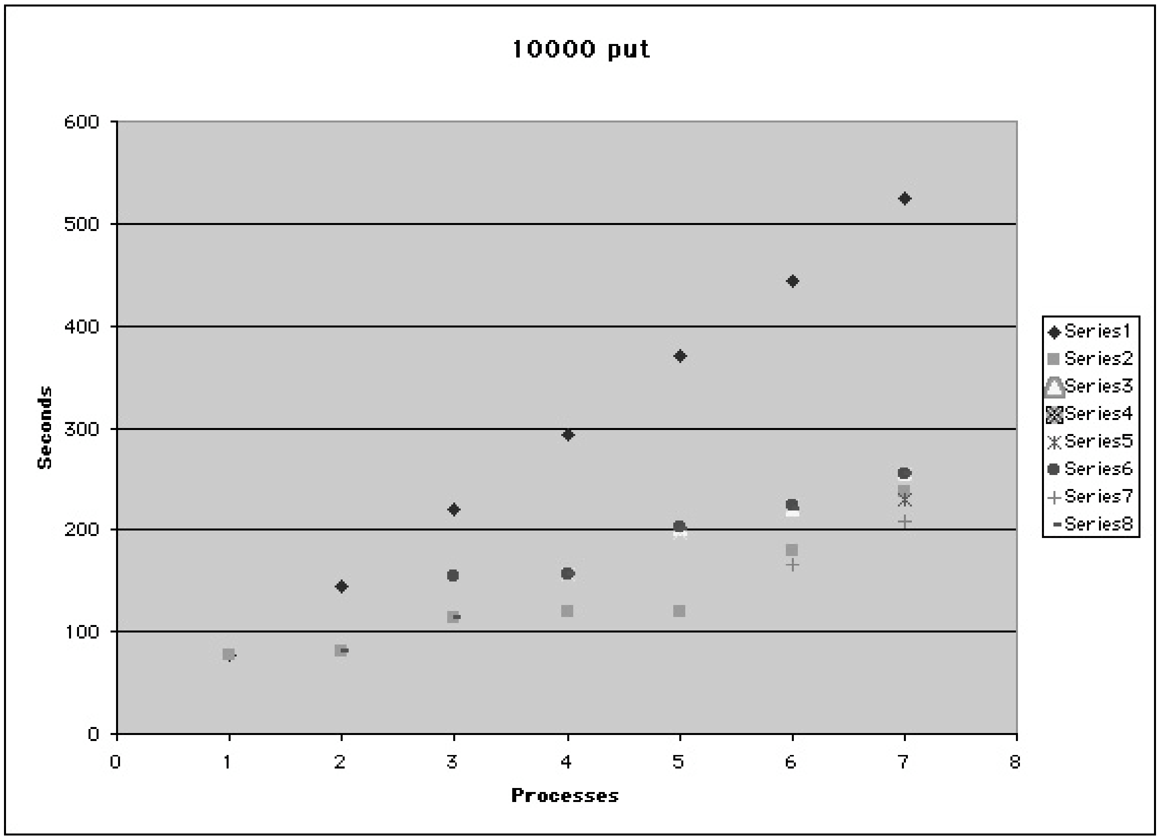


Fig. 2D – Benchmark on Parallelism of Server-Server Communication (with disk access as in Fig. 2A)

Series 2 through 8 were obtained from running the processes on different Access Servers against one Library Server, each series goes with one server. In this experiment, the processes did not finish at the same time, which most likely is due to disk-caching on the Library Server – `lond`-children whose datafile was (partly) in disk cache finished earlier. With seven processes from seven different servers, the operation took 255 seconds till the last process was finished for 70,000 `put` commands (270 per second) – versus 530 seconds if the processes ran on the same server (130 per second).

Dynamic Resource Replication

Since resources are assembled into higher order resources simply by reference, in principle it would be sufficient to retrieve them from the respective Home Servers of the authors. However, there are several problems with this simple approach: since the resource assembly mechanism is designed to facilitate content assembly from a large number of widely distributed sources, individual sessions would depend on a large number of machines and network connections to be available, thus be rather fragile. Also, frequently accessed resources could potentially drive individual machines in the network into overload situations.

Finally, since most resources depend on content handlers on the Access Servers to be served to a client within the session context, the raw source would first have to be transferred across the Network from the respective Library Server to the Access Server, processed there, and then transferred on to the client.

To enable resource assembly in a reliable and scalable way, a dynamic resource replication scheme was developed. **Fig. 1.1.3** shows the details of this mechanism.

Anytime a resource out of the resource space is requested, a handler routine is called which in turn calls the replication routine (**Fig. 1.1.3A**). As a first step, this routines determines whether or not the resource is currently in replication transfer (**Fig. 1.1.3A, Step D1a**). During replication transfer, the incoming data is stored in a temporary file, and **Step D1a** checks for the presence of that file. If transfer of a resource is actively going on, the controlling handler receives an error message, waits for a few seconds, and then calls the replication routine again. If the resource is still in transfer, the client will receive the message "Service currently not available".

In the next step (**Fig. 1.1.3A, Step D1b**), the replication routine checks if the URL is locally present. If it is, the replication routine returns OK to the controlling handler, which in turn passes the request on to the next handler in the chain.

If the resource is not locally present, the Home Server of the resource author (as extracted from the URL) is determined (**Fig. 1.1.3A, Step D2**). This is done by contacting all library servers in the author's domain (as determined from the lookup table, see **Fig. 1.1.2B**). In **Step D2b** a query is sent to the remote server whether or not it is the Home Server of the author (in our current implementation, an additional cache is used to store already identified Home Servers (not shown in the figure)). In Step **D2c**, the remote server answers the query with True or False. If the Home Server was found, the routine continues, otherwise it contacts the next server (**Step D2a**). If no server could be found, a "File not Found" error message is issued. In our current implementation, in this step the Home Server is also written into a cache for faster access if resources by the same author are needed again (not shown in the figure).

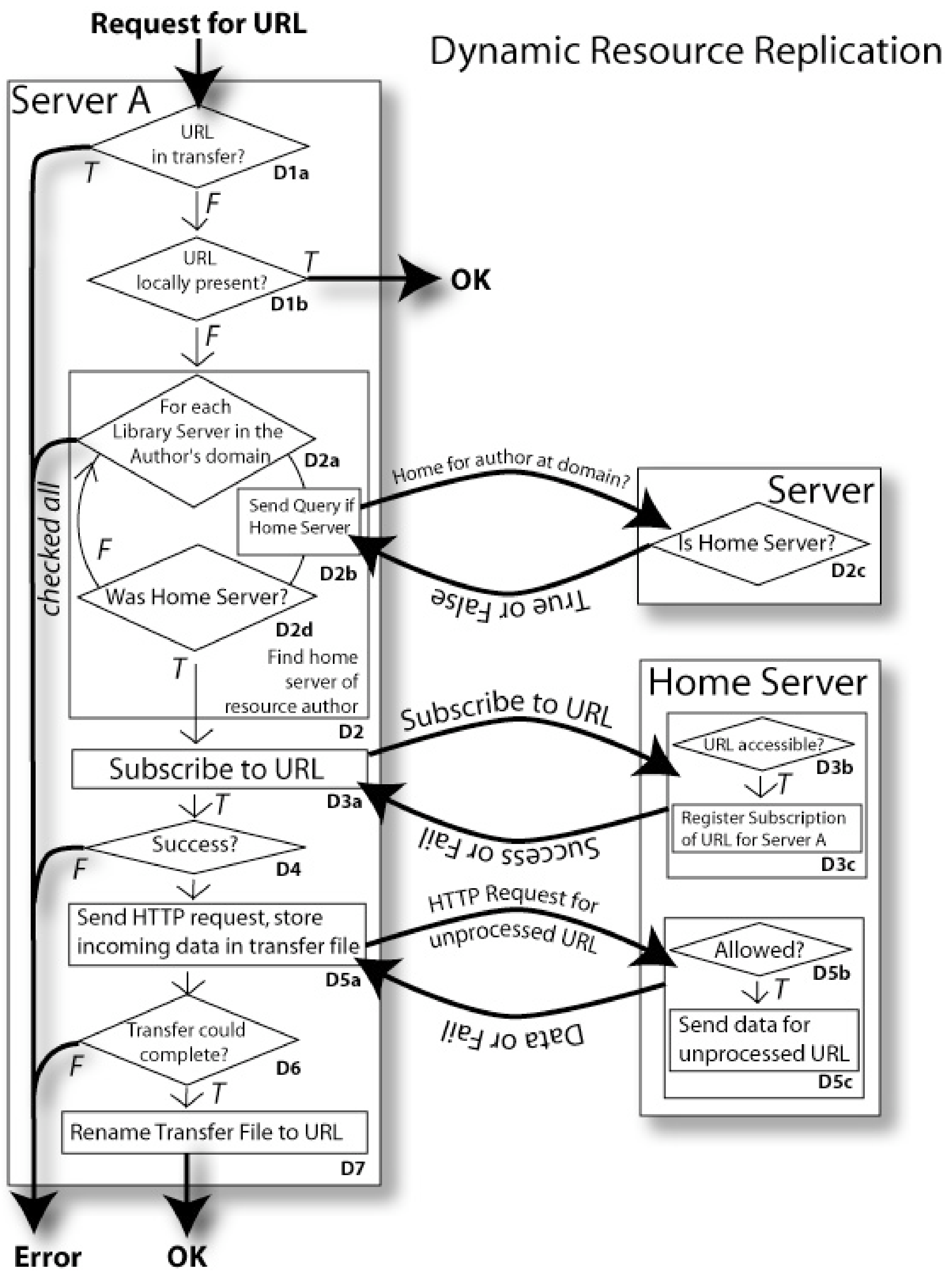

Request for URL
Dynamic Resource Replication
Server A
URL in transfer?
T
D1a
F
URL locally present?
T
D1b
OK
F
For each Library Server in the Author's domain
D2a
Send Query if Home Server
Home for author at domain?
Server
Is Home Server?
D2c
True or False
F
checked all
D2b
Was Home Server?
D2d
T
Find home server of resource author
D2
Subscribe to URL
Home Server
URL accessible?
D3b
T
Register Subscription of URL for Server A
D3c
Subscribe to URL
T
D3a
Success or Fail
Success?
F
D4
Send HTTP request, store incoming data in transfer file
D5a
HTTP Request for unprocessed URL
Allowed?
D5b
T
Send data for unprocessed URL
D5c
Data or Fail
Transfer could complete?
F
D6
T
Rename Transfer File to URL
D7
Error
OK

Fig. 1.1.3A – Dynamic Resource Replication, subscription

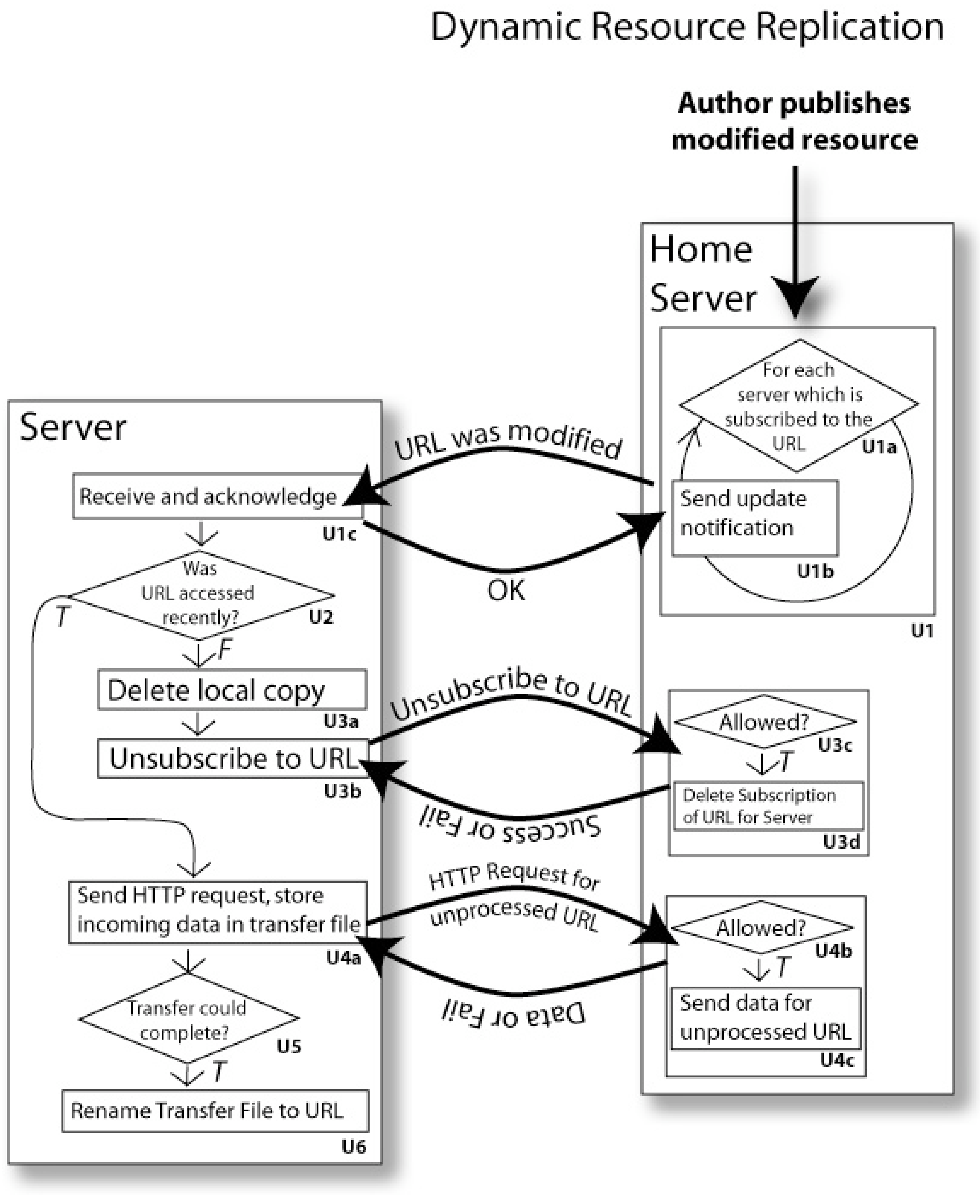

Dynamic Resource Replication
Author publishes modified resource
Home Server
For each server which is subscribed to the URL
U1a
Send update notification
U1b
U1
Server
URL was modified
Receive and acknowledge
U1c
OK
Was URL accessed recently?
U2
T
F
Delete local copy
U3a
Unsubscribe to URL
Unsubscribe to URL
U3b
Allowed?
U3c
T
Delete Subscription of URL for Server
U3d
Success or Fail
HTTP Request for unprocessed URL
Send HTTP request, store incoming data in transfer file
U4a
Allowed?
U4b
T
Send data for unprocessed URL
U4c
Data or Fail
Transfer could complete?
U5
T
Rename Transfer File to URL
U6

Fig. 1.1.3B – Dynamic Resource Replication, modification

In **Step D3a**, the routine sends a subscribe command for the URL to the Home Server of the author. The Home Server first determines if the resource is present, and if the access privileges allow it to be copied to the requesting server (**Fig. 1.1.3A, Step D3b**). If this is true, the requesting server is added to the list of subscribed servers for that resource (**Step D3c**). The Home Server will reply with either OK or an error message, which is determined in **Step D4**. If the remote resource was not present, the error message "File not Found" will be passed on to the client, if the access was not allowed, the error message "Access

Denied" is passed on. If the operation succeeded, the requesting server sends an HTTP request for the resource out of the `/raw` server content resource area of the Home Server.

The Home Server will then check if the requesting server is part of the network, and if it is subscribed to the resource (**Step D5b**). If it is, it will send the resource via HTTP to the requesting server without any content handlers processing it (**Step D5c**). The requesting server will store the incoming data in a temporary data file (**Step D5a**) – this is the file that **Step D1a** checks for. If the transfer could not complete, and appropriate error message is sent to the client (**Step D6**). Otherwise, the transferred temporary file is renamed as the actual resource, and the replication routine returns OK to the controlling handler (**Step D7**).

Fig. 1.1.3B depicts the process of modifying a resource. When an author publishes a new version of a resource, the Home Server will contact every server currently subscribed to the resource (**Fig. 1.1.3B, Step U1**), as determined from the list of subscribed servers for the resource generated in **Fig. 1.1. 3A, Step D3c**. The subscribing servers will receive and acknowledge the update message (**Step U1c**). The update mechanism finishes when the last subscribed server has been contacted (messages to unreachable servers are buffered).

Each subscribing server will check if the resource in question had been accessed recently, that is, within a configurable amount of time (**Step U2**).

If the resource had not been accessed recently, the local copy of the resource is deleted (**Step U3a**) and an unsubscribe command is sent to the Home Server (**Step U3b**). The Home Server will check if the server had indeed originally subscribed to the resource (**Step U3c**) and then delete the server from the list of subscribed servers for the resource (**Step U3d**).

If the resource had been accessed recently, the modified resource will be copied over using the same mechanism as in **Step D5a** through **D7** of **Fig. 1.1.3A** (**Fig. 1.1.3B**, **Steps U4a** through **U6**).

Load Balancing

`lond` provides a function to query the server's current `loadavg`. As a configuration parameter, one can determine the value of `loadavg`, which is to be considered 100%, for example, 2.00.

Access servers can have a list of spare access servers, `/home/httpd/lonTabs/spares.tab`, to offload sessions depending on own workload. This check happens is done by the login handler. It re-directs the login information and session to the least busy spare server if itself is overloaded. An additional round-robin IP scheme possible. See **Fig. 1.1.4** for an example of a load-balancing scheme.

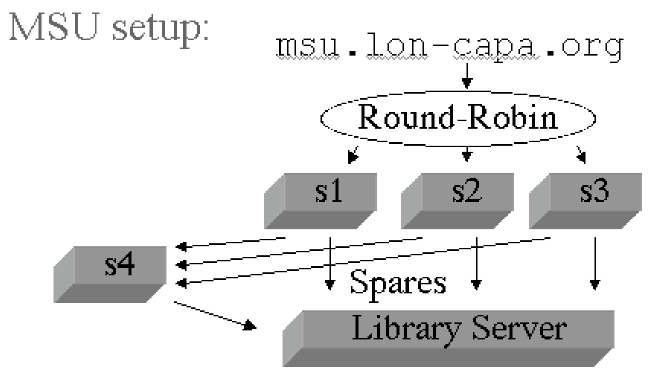


Fig. 1.1.4 – Example of Load Balancing

Session Two: Apache Handlers (loncapa_apache.conf), Authentication and Access Control, Session Environment (Matthew)

Server Content Resource Areas

Internally, all resources are identified primarily by their URL. Different logical areas of the server are distinguished by the beginning part of the URL:

- `/adm`: publicly available content, logos, manual pages, etc.

- `/res/`*`domainname`*`/`*`authorname`*`/..`: the resource area, holding course maps, HTML pages, homework, movies, applets, etc. Access to these files is restricted by the cookie-based authentication mechanism. Content in this area will be served by type-dependent handlers, for example, one handlers to serve homework problems, and another one for TeX resources. The structure of this area of the server is exactly the same on every server, even though not all resources might be present everywhere.

- `/raw/`*`domainname`*`/`*`authorname`*`/..`: internally, this is just a symbolic link to the `res` directory, however, no content handlers are called when serving a resource and access is controlled by IP rather than cookies. This structure is used for replication of resources between servers.

- `/~`*`authorname`*`/.., /priv/`*`authorname`*: the content construction space. This is normal UNIX filespace, which however can only by viewed on the web by the authors themselves through the cookie based authentication. Content handlers are active for this space. This space can be mounted on other UNIX machines, as well as AppleShare and Windows. Below the *`authorname`*, this directory has the same structure as the resource space of the author.

- `/lon-status/..`: LON-CAPA status information – behind basic http authentication so it is not dependent on system functions

Authors can only write-access the `/~`*`authorname`*`/` space. They can copy resources into the resource area through the publication step, and move them back through a retrieve step. Authors do not have direct write-access to their resource space.

Apache Request Cycle and Handlers

The standard mode in which the Apache web server is used is that a URL corresponds to some static file on the file system, which then more or less gets sent out as-is. Slight deviations from that simple principle are however already the directory listing function, virtual servers, and the cgi-bin directory. In the latter case, Apache executes the file in a new process and provides both the input to the process in the environment, and routes the output to the client. Other deviations are the error messages.

In a more general view, URLs to Apache are URIs (Uniform Resource Identifiers), which may are may not correspond to a physical file, which in turn may or may not be sent out as-is.

As a request for a URI gets sent to the Apache server, it goes through several phases. At each phase ("stage") along the process, handler modules can be defined to deal with the request. Details about these phases are given in the Eagle book Chapter 3 (page 60).

These handler modules are not like cgi-scripts executed in a separate process, but are dynamically linked into the Apache child processes themselves – they run inside of Apache. The mod_perl mechanism in addition links the Perl interpreter into the Apache child process. Modules are pre-interpreted ("compiled") by the Perl interpreter when they are first loaded, and the pre-interpreted code stays permanently in the memory allocated to that Apache child process. The result is a significant speed-up, and through the flexible mechanism of module registration and different stages of the process, a high degree of customizability.

LON-CAPA does not use Apache::Registry, and so in addition avoids the unnecessary overhead of emulating a cgi-environment within the handlers. Instead, it makes direct use of the Apache Perl API (Chapter 9, Eagle book).

Handler Definition

Which chain of handler deals with which kind of resource is defined in `/etc/httdp/conf/loncapa_apache.conf`. LON-CAPA only defines handlers at the (in order) header-parser, access control, and response phase (figure 3-3 Eagle book). In the header-parser phase, the replication handler is run, and in the access-control phase, the various access-handlers. Future handlers will mostly be implemented in the response phase. **Fig. 1.2.2** shows excerpts of the configuration script `/etc/httpd/loncapa_apache.conf` for these modules.

```
# ------------------------------------------------------------ Access Handlers

<LocationMatch "^/res.*">
PerlAccessHandler       Apache::lonacc
PerlHeaderParserHandler Apache::lonrep
ErrorDocument     403 /adm/login
ErrorDocument     404 /adm/notfound.html
ErrorDocument     406 /adm/roles
ErrorDocument     500 /adm/errorhandler
</LocationMatch>

<LocationMatch "^/priv.*">
PerlAccessHandler Apache::loncacc
SetHandler        perl-script
PerlHandler       Apache::lonconstruct
ErrorDocument     403 /adm/login
ErrorDocument     404 /adm/notfound.html
ErrorDocument     406 /adm/unauthorized.html
ErrorDocument     500 /adm/errorhandler
</LocationMatch>

<LocationMatch "^/raw.*">
PerlAccessHandler Apache::lonracc
</LocationMatch>

<LocationMatch "^/\~.*">
PerlAccessHandler Apache::loncacc
ErrorDocument     403 /adm/login
ErrorDocument     404 /adm/notfound.html
ErrorDocument     406 /adm/unauthorized.html
ErrorDocument     500 /adm/errorhandler
AllowOverride None
</LocationMatch>

…
# -------------------------------------------- Resource Space Content Handlers
```

```
<LocationMatch "^/res/.*/$">
SetHandler perl-script
PerlHandler Apache::lonindexer
</LocationMatch>

<LocationMatch "^/res/.*\.tex$">
SetHandler perl-script
PerlHandler Apache::lontex
</LocationMatch>

<LocationMatch "^/res/.*\.page$>
SetHandler perl-script
PerlHandler Apache::lonpage
</LocationMatch>

...
<LocationMatch "^/(res|\~).*\.(xml|html|htm|xhtml|xhtm)$">
SetHandler perl-script
PerlHandler Apache::lonxml
</LocationMatch>

<LocationMatch "^/(res|\~).*\.(problem|exam|quiz|assess|survey|form)$">
SetHandler perl-script
PerlHandler Apache::lonhomework
</LocationMatch>


# -------------------------------------------------------------- Admin Programs

<Location /adm/roles>
PerlAccessHandler       Apache::lonacc
SetHandler perl-script
PerlHandler Apache::lonroles
ErrorDocument     403 /adm/login
ErrorDocument     500 /adm/errorhandler
</Location>

<Location /adm/login>
SetHandler perl-script
PerlHandler Apache::lonlogin
</Location>

...
<Location /adm/annotations>
PerlAccessHandler       Apache::lonacc
SetHandler perl-script
PerlHandler Apache::admannotations
ErrorDocument     403 /adm/login
ErrorDocument     500 /adm/errorhandler
</Location>
... etc ...
```

Fig. 1.2.2 – Excerpts of `loncapa_apache.conf`

Authentication Overview

A user can log into any server in the network and run sessions. The server responsible for initially authenticating the user is the user's homeserver.

When a user first accesses a server within a browser session, he or she is challenged to provide authentication information in the form of username, password and domain – this is done by `lonlogin`, which is the `error_document` for `lonacc` and `loncacc` (the normal authentication handlers). When the server receives that information, it asks all library servers in the domain that the user specified to validate the information – this is done by `lonauth`.

The user's home server will answer with `authorized` or `non_authorized`, all others with `unknown_user`. If one server authorizes the user, a cookie is returned to the user by `lonauth` and the session is initialized on the local server by establishing the session environment file. If a server sends `non_authorized`, the user is rejected. **Fig. 1.3.1** illustrates this process.

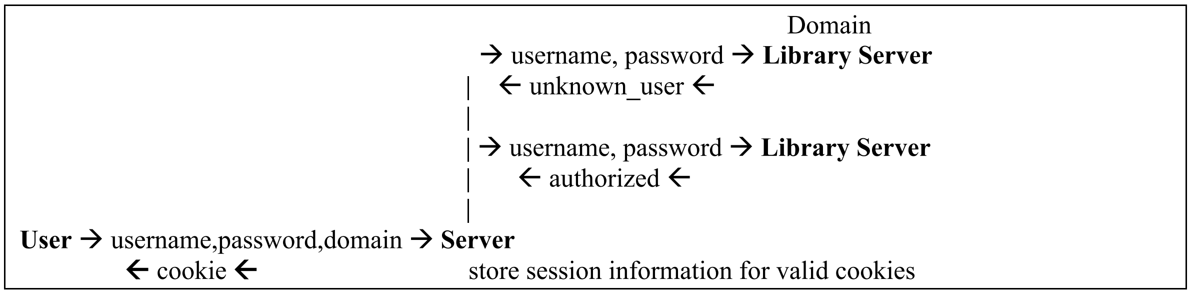

Domain
→ username, password → **Library Server**
← unknown_user ←
→ username, password → **Library Server**
← authorized ←
User → username,password,domain → **Server**
← cookie ←
store session information for valid cookies

Fig. 1.3.1 – Illustration of Authentication Mechanism

At all subsequent interactions, the client sends the cookie back to the server – if the cookie is missing or invalid, the user is re-challenged for login information. Handlers are `lonacc` and `loncacc`. Cookies expire by closing the browser and are invalidated when the user logs out or logs in a second time into the same machine from another browser.

Authentication Mechanisms, User Data, Passwords

On the library servers, it is a routine in `lond` that does the authentication. It checks if this is the user's homeserver, checks the password, and answers with `"unknown_user", "authorized"` or `"non_authorized".`

Determination if this is the user's homeserver is done by the presence of his or her password file in

```
/home/httpd/lonUsers/domain/1.char/2.char/3.char/username/passwd
```

for example

```
/home/httpd/lonUsers/msu/s/m/i/smith/passwd
```

The password is stored in the format `mechanism:info,` where mechanism can currently be `unix`, `krb4` or `internal`. For `krb4`, the `info` is the Kerberos domain, for `internal` it is the crypt password itself. `unix` simply authenticates against `/etc/passwd`.

Environment

The access handlers – coming early in the Apache request cycle – also set up the session environment. The cookie received from the web client is a pointer to the session profiles, which are stored in a directory for temporary files `(/home/httpd/lonIDs/)`.

Standard Components

These are the standard components of the environment added by Apache and the shell.

```
AUTH_TYPE ---- Basic
DOCUMENT_ROOT ---- /home/httpd/html
GATEWAY_INTERFACE ---- CGI-Perl/1.1
HTTP_ACCEPT ---- image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
HTTP_ACCEPT_CHARSET ---- iso-8859-1,*,utf-8
HTTP_ACCEPT_ENCODING ---- gzip
HTTP_ACCEPT_LANGUAGE ---- en,pdf
HTTP_CONNECTION ---- Keep-Alive
HTTP_COOKIE ---- SITESERVER=ID=cbc6695505253a2ff0e7bb7110574d90;
lonID=kortemey_990461714_msu_msul1
HTTP_HOST ---- zaphod.lite.msu.edu
HTTP_REFERER ----
HTTP_USER_AGENT ---- Mozilla/4.75 (Macintosh; U; PPC)
MOD_PERL ---- mod_perl/1.21
PATH ---- /sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bi€
QUERY_STRING ----
REMOTE_ADDR ---- 35.8.63.7
REMOTE_PORT ---- 1844
REMOTE_USER ---- lonadm
REQUEST_METHOD ---- GET
REQUEST_URI ---- /adm/test
SCRIPT_FILENAME ---- /home/httpd/html/adm/test
SCRIPT_NAME ---- /adm/test
SERVER_ADDR ---- 35.8.63.51
SERVER_ADMIN ---- korte@lite.msu.edu
SERVER_NAME ---- zaphod.lite.msu.edu
SERVER_PORT ---- 80
SERVER_PROTOCOL ---- HTTP/1.0
SERVER_SIGNATURE ----
SERVER_SOFTWARE ---- Apache/1.3.9 (Unix) (Red Hat/Linux) mod_perl/1.21
```

Resource Access Control

The following values are set by traceroute at the initialization of the course and are used by access handlers to check if a resource can be served to a user.

```
acc.cond.msu_12679c3ed543a25msul1.0 ---- 0
acc.cond.msu_12679c3ed543a25msul1.1 ---- 0
acc.res.msu_12679c3ed543a25msul1. ---- &:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp ---- &welcome267.htm:1&welcomelbs267.htm:0&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/applist/Spectrum ---- &s.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/applist/chain ---- &chain.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/applist/coulomb ---- &orbit.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/applist/induct ---- &faraday.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/conversions ----
&conv_area.htm:1&prefixes.htm:1&conv_power.htm:1&intro.htm:1&conv_temperature.htm:1&conv_
time.htm:1&conv_velocity.htm:1&conversions.sequence:1&conv_length.htm:1&conv_mass.htm:1&c
onv_pressure.htm:1&conv_volume.htm:1&conv_energy.htm:1&conv_angle.htm:1&sibaseunits.htm:1
&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap13 ---- &cd371.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap16 ----
&cd427.htm:1&cd424.htm:1&Stable.htm:1&cd421.htm:1&Constants.htm:1&cd425.htm:1&cd422.htm:1
&kap16.sequence:1&cd426.htm:1&cd423.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap17 ----
&cd436.htm:1&kap17.sequence:1&cd433.htm:1&cd430.htm:1&cd437.htm:1&cd434.htm:1&cd428.htm:1
&cd431.htm:1&cd435.htm:1&cd429.htm:1&geometry.htm:1&cd432.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap17/calculus ---- &calc.htm:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap18 ----
&RR440.htm:1&RR449b.htm:1&RR446b.htm:1&cd473.htm:1&RR453.htm:1&RR447.htm:1&RR443b.htm:1&R
R449a.htm:1&RR450.htm:1&RR452a.htm:1&RR444.htm:1&RR446a.htm:1&sum18a.htm:1|1&RR441.htm:1&
RR443a.htm:1&eField.htm:1&RR454.htm:1&RR444b.htm:1&RR448.htm:1&RR447app.htm:1&RR451.htm:1
&RR445.htm:1&RR453a.htm:1&kap18.sequence:1&RR439.htm:1&RR447a.htm:1&RR442.htm:1&RR450a.ht
m:1&RR444a.htm:1&RR448b.htm:1&RR445newch.htm:1&RR4460app.htm:1&RR455.htm:1&RR441a.htm:1&R
R449.htm:1&RR452.htm:1&cd438.htm:1&RR446.htm:1&RR454a.htm:1&sum18.htm:1|1&RR446c.htm:1&RR
448a.htm:1&RR451a.htm:1&RR443.htm:1&RR445answer.htm:0&RR445a.htm:1&
```

```
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap18/demo ----
&vid440.htm:0&vid440sm.htm:0&vid449.htm:0&vid449-
a.htm:0&vid441.htm:0&vid449sm.htm:0&vid455.htm:0&egun.htm:0&egunsm.htm:0&vid441sm.htm:0&v
id449-asm.htm:0&vid455sm.htm:0&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap18/problems ----
&cd460b.problem:1&cd458.problem:1&cd462.problem:1&cd457.problem:1&cd461.problem:1&cd460.p
roblem:1&cd464.problem:1&cd459.problem:1&cd463.problem:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap18a ---- &kap18a.sequence:1&
acc.res.msu_12679c3ed543a25msul1.msu/mmp/kap18a/problems ----
&cd465.problem:1&cd472.problem:1&cd466.problem:1&cd467.problem:1&cd470.problem:1&cd468.pr
oblem:1&cd471.problem:1&cd469.problem:1&
```

… etc for all resources in the course …

Browser Information

```
browser.mathml ----
browser.os ---- mac
browser.type ---- netscape
browser.version ---- 4.75
```

Cached Information about Courses and their Description

```
course.msu_12679c3ed543a16msul1.description ---- lbs267L Lab SS01
course.msu_12679c3ed543a16msul1.domain ---- msu
course.msu_12679c3ed543a16msul1.home ---- msul1
course.msu_12679c3ed543a16msul1.last_cache ---- 990461725
course.msu_12679c3ed543a16msul1.num ---- 12679c3ed543a16msul1
course.msu_12679c3ed543a16msul1.url ---- msu/mmp/lbs267l.sequence
course.msu_12679c3ed543a25msul1.description ---- lbs267 Lecture SS01
course.msu_12679c3ed543a25msul1.domain ---- msu
course.msu_12679c3ed543a25msul1.home ---- msul1
course.msu_12679c3ed543a25msul1.last_cache ---- 990461728
course.msu_12679c3ed543a25msul1.num ---- 12679c3ed543a25msul1
course.msu_12679c3ed543a25msul1.url ---- msu/mmp/lbs267.sequence
course.msu_12679c3ed543a37msul1.description ---- Demo Course
course.msu_12679c3ed543a37msul1.domain ---- msu
course.msu_12679c3ed543a37msul1.home ---- msul1
course.msu_12679c3ed543a37msul1.last_cache ---- 990461725
course.msu_12679c3ed543a37msul1.num ---- 12679c3ed543a37msul1
course.msu_12679c3ed543a37msul1.url ---- msu/korte/demo.sequence
```

Information Imported from the Environment Database File of the User

```
environment.favorite.cake ---- Cheese Cake
environment.favorite.color ---- green
environment.firstname ---- Gerd
environment.id ---- z12345678
environment.lastname ---- Kortemeyer
```

Information about the Request

```
httpref./res/msu/mmp/ ---- /res/msu/mmp/welcome267.htm
request.ambiguous ---- adm/pages/index.html
request.course.fn ---- /home/httpd/perl/tmp/kortemey_msu_12679c3ed543a25msul1
request.course.id ---- msu_12679c3ed543a25msul1
request.course.sec ----
request.course.uri ---- msu/mmp/lbs267.sequence
request.filename ---- /home/httpd/html/adm/test
request.host ---- zaphod.lite.msu.edu
request.role ---- cc./msu/12679c3ed543a25msul1
request.state ---- published
```

Information about the User

```
user.domain ---- msu
user.environment ---- /home/httpd/lonIDs/kortemey_990461714_msu_msul1.id
user.home ---- msul1
user.login.time ---- 990461714
user.name ---- kortemey
```

Information about User Roles and Privileges

```
user.priv.au./msu/./ ---- :sma&F:gan&F
user.priv.au./msu/./msu/ ---- :cca&IK:are&F:bre&F:cre&F:ere&F
user.priv.ca./msu/korte./ ---- :sma&F:gan&F
user.priv.ca./msu/korte./msu/ ---- :are&F:bre&F:cre&F:ere&F
user.priv.ca./msu/korte./msu/korte ----
user.priv.cc./msu/12679c3ed543a16msul1./ ---- :sma&F:bre&F:mcr&F
user.priv.cc./msu/12679c3ed543a16msul1./msu/ ----
user.priv.cc./msu/12679c3ed543a16msul1./msu/12679c3ed543a16msul1 ----
:opa&F:srm&F:gan&F:are&F:ccr&IK:cep&IK:cta&IK:cre&F:cst&IK:cin&IK:ere&F:vgr&F
user.priv.cc./msu/12679c3ed543a25msul1./ ---- :sma&F:bre&F:mcr&F
user.priv.cc./msu/12679c3ed543a25msul1./msu/ ----
user.priv.cc./msu/12679c3ed543a25msul1./msu/12679c3ed543a25msul1 ----
:opa&F:srm&F:gan&F:cta&IK:cep&IK:ccr&IK:are&F:cin&IK:cst&IK:cre&F:ere&F:vgr&F
user.priv.cc./msu/12679c3ed543a37msul1./ ---- :sma&F:mcr&F:bre&F
user.priv.cc./msu/12679c3ed543a37msul1./msu/ ----
user.priv.cc./msu/12679c3ed543a37msul1./msu/12679c3ed543a37msul1 ----
:opa&F:srm&F:gan&F:are&F:ccr&IK:cep&IK:cta&IK:cre&F:cst&IK:cin&IK:ere&F:vgr&F
user.priv.cm./ ---- :sma&F:mcr&F:gan&F:bre&F
user.priv.cm./msu/ ----
:mau&F:cca&IK:cad&UIK:ccc&U:cst&UIK:cdg&UIK:are&F:cli&UIK:cta&UIK:cep&UIK:ccr&UIK:bre&F:c
au&U:cre&F:cin&UIK:ere&F
user.priv.cm./msu/12679c3ed543a16msul1 ----
:opa&F:srm&F:gan&F:are&F:ccr&IK:cep&IK:cta&IK:cre&F:cst&IK:cin&IK:ere&F:vgr&F
user.priv.cm./msu/12679c3ed543a25msul1 ----
:opa&F:srm&F:gan&F:are&F:ccr&IK:cep&IK:cta&IK:cre&F:cst&IK:cin&IK:ere&F:vgr&F
user.priv.cm./msu/12679c3ed543a37msul1 ----
:opa&F:cst&IK:vgr&F:srm&F:gan&F:are&F:ccr&IK:cep&IK:cta&IK:cre&F:cin&IK:ere&F
user.priv.cm./msu/korte ----
user.priv.dc./msu/./ ---- :sma&F
user.priv.dc./msu/./msu/ ----
:mau&F:cad&UIK:ccr&UIK:cep&UIK:cta&UIK:cli&UIK:ccc&U:cau&U:cst&UIK:cin&UIK:cdg&UIK
user.role.au./msu/ ---- 964531839.0
user.role.ca./msu/korte ---- .
user.role.cc./msu/12679c3ed543a16msul1 ---- 964531839.0
user.role.cc./msu/12679c3ed543a25msul1 ---- 964531839.0
user.role.cc./msu/12679c3ed543a37msul1 ---- 964531839.0
user.role.dc./msu/ ---- 964531839.0
```

Handler Reference: LON-CAPA and the 77 Web Perl Modules

Scott Harrison, freeware volunteer, sharrison@sourceforge.net

LON-CAPA provides many different web services for coordinating online educational interactions. Currently, these web services are made available by 77 different perl modules. The invocation of these perl modules is many times due to the URI format based on entries inside `/etc/httpd/conf/loncapa_apache.conf`.

Throughout this technical manual, various web perl modules are described in detail. Here is a summary of all the 77 web perl modules:

Name/Location	Description
SOURCE: rat/lonwrapper.pm TARGET: home/httpd/lib/perl/Apache/lonwrapper.pm	Wrapper for external and binary files as standalone resources. Edit handler for rat maps; TeX content handler.

SOURCE: loncom/publisher/loncfile.pm TARGET: home/httpd/lib/perl/Apache/loncfile.pm	Provides web-based functionality for file copy, rename, mkdir, etc, in the construction space menu.
SOURCE: loncom/interface/lonstatistics.pm TARGET: home/httpd/lib/perl/Apache/lonstatistics.pm	Handler to show statistics on solving LON-CAPA problems.
SOURCE: loncom/publisher/londiff.pm TARGET: home/httpd/lib/perl/Apache/londiff.pm	Handler to show difference between two files.
SOURCE: loncom/publisher/lonupload.pm TARGET: home/httpd/lib/perl/Apache/lonupload.pm	Handler to upload files through browser into construction space.
SOURCE: loncom/homework/essayresponse.pm TARGET: home/httpd/lib/perl/Apache/essayresponse.pm	Handler to evaluate essay (ungraded) style responses.
SOURCE: loncom/homework/externalresponse.pm TARGET: home/httpd/lib/perl/Apache/externalresponse.pm	Handler to evaluate externally graded responses.
SOURCE: loncom/homework/loncapagrade.pm TARGET: home/httpd/lib/perl/Apache/loncapagrade.pm	Handler to evaluate externally graded responses.
SOURCE: loncom/publisher/lonpubdir.pm TARGET: home/httpd/lib/perl/Apache/lonpubdir.pm	Handler to publish directories.
SOURCE: loncom/publisher/lonretrieve.pm TARGET: home/httpd/lib/perl/Apache/lonretrieve.pm	Handler to retrieve old versions from resource space.
SOURCE: loncom/homework/edit.pm TARGET: home/httpd/lib/perl/Apache/edit.pm	Helper functions when in homework edit mode.
SOURCE: loncom/interface/lonmeta.pm TARGET: home/httpd/lib/perl/Apache/lonmeta.pm	Metadata display handler.
SOURCE: rat/lonambiguous.pm TARGET: home/httpd/lib/perl/Apache/lonambiguous.pm	Handler to resolve ambiguous file locations.
SOURCE: rat/lonratparms.pm TARGET: home/httpd/lib/perl/Apache/lonratparms.pm	Handler to set resource parameters inside of the RAT based on metadata.
SOURCE: rat/lonsequence.pm TARGET: home/httpd/lib/perl/Apache/lonsequence.pm	Handler for showing sequence objects of educational resources.
SOURCE: loncom/interface/loncreatecourse.pm TARGET: home/httpd/lib/perl/Apache/loncreatecourse.pm	Allows domain coordinators to create new courses and assign course coordinators.
SOURCE: loncom/interface/loncreateuser.pm TARGET: home/httpd/lib/perl/Apache/loncreateuser.pm	Allows users to within their own privileges create/edit users and give them roles.
SOURCE: loncom/interface/lonchart.pm TARGET: home/httpd/lib/perl/Apache/lonchart.pm	Produces simple LectureOnline-like student assessment performance chart

SOURCE: loncom/interface/loncommon.pm TARGET: home/httpd/lib/perl/Apache/loncommon.pm	Makes a table out of the previous attempts. Inputs result_from_symbread, user, domain, home_server, course_id.
SOURCE: loncom/homework/grades.pm TARGET: home/httpd/lib/perl/Apache/grades.pm	Handles the viewing of grades.
SOURCE: loncom/homework/imageresponse.pm TARGET: home/httpd/lib/perl/Apache/imageresponse.pm	Coordinates the response to clicking an image.
SOURCE: loncom/homework/optionresponse.pm TARGET: home/httpd/lib/perl/Apache/optionresponse.pm	Handles tags associated with showing a list of options.
SOURCE: loncom/homework/outputtags.pm TARGET: home/httpd/lib/perl/Apache/outputtags.pm	Handles tags associated with output. Seems to relate to due dates of the assignment.
SOURCE: loncom/interface/lontest.pm TARGET: home/httpd/lib/perl/Apache/lontest.pm	Used for debugging and testing the LON-CAPA system.
SOURCE: loncom/homework/radiobuttonresponse.pm TARGET: home/httpd/lib/perl/Apache/radiobuttonresponse.pm	Handles multiple-choice style responses.
SOURCE: loncom/interface/lonassignments.pm TARGET: home/httpd/lib/perl/Apache/lonassignments.pm	Handles processing of assignments.
SOURCE: loncom/interface/loncommunicate.pm TARGET: home/httpd/lib/perl/Apache/loncommunicate.pm	Will be the access handler to email sending, as well as the planned chatrooms, etc.
SOURCE: loncom/interface/lonerrorhandler.pm TARGET: home/httpd/lib/perl/Apache/lonerrorhandler.pm	Handles errors.
SOURCE: loncom/interface/lonevaluate.pm TARGET: home/httpd/lib/perl/Apache/lonevaluate.pm	Handles evaluation.
SOURCE: loncom/interface/lonfeedback.pm TARGET: home/httpd/lib/perl/Apache/lonfeedback.pm	Handles feedback from students to instructors and system administrators. Provides a screenshot of the current resource, as well as previous attempts if the resource was a homework. Used by lonmsg.pm.
SOURCE: loncom/interface/lonnavmaps.pm TARGET: home/httpd/lib/perl/Apache/lonnavmaps.pm	Handles navigational maps.
SOURCE: loncom/interface/lonpreferences.pm TARGET: home/httpd/lib/perl/Apache/lonpreferences.pm	Handles user preferences associated with customizing the online LON-CAPA educational environment.
SOURCE: loncom/interface/lonprintout.pm TARGET: home/httpd/lib/perl/Apache/lonprintout.pm	Handles the production of printable files and resources.
SOURCE: loncom/interface/lonsearchcat.pm TARGET: home/httpd/lib/perl/Apache/lonsearchcat.pm	Handles a searchable catalogue.

SOURCE: loncom/interface/londropadd.pm TARGET: home/httpd/lib/perl/Apache/londropadd.pm	Allows course coordinators to upload courselists in different formats, and automatically create users (if they do not exist already), assign them the role of student in a course, and add them to the classlist.
SOURCE: loncom/interface/lonmsg.pm TARGET: home/httpd/lib/perl/Apache/lonmsg.pm	lonmsg.pm has several functions to send and receive internal messages. author_res_msg - send message to resource author. user_crit_msg - send a critical message to a user. A critical message will require acknowledgment by the recipient and the sender will be notified. user_crit_received - routine to trigger acknowledgment. statuschange - change the status of a message (read, replied, forwarded, etc). The handler also displays messages, has routines to reply, etc.
SOURCE: loncom/homework/hint.pm TARGET: home/httpd/lib/perl/Apache/hint.pm	This handler coordinates the delivery of hints to students working on LON-CAPA problems and assignments.
SOURCE: loncom/interface/lonspreadsheet.pm TARGET: home/httpd/lib/perl/Apache/lonspreadsheet.pm	Spreadsheets are completely web-based. They exist on the level of a whole course, a student, and individual assessments.
SOURCE: loncom/interface/lonparmset.pm TARGET: home/httpd/lib/perl/Apache/lonparmset.pm	Handler to resolve ambiguous file locations
SOURCE: loncom/publisher/lonconstruct.pm TARGET: home/httpd/lib/perl/Apache/lonconstruct.pm	Page wrapper for handling construction space.
SOURCE: loncom/publisher/lonpublisher.pm TARGET: home/httpd/lib/perl/Apache/lonpublisher.pm	Publishes an LON-CAPA educational resource complete with metadata (authorship, language, copyright, creation date, etc).
SOURCE: loncom/interface/lonmenu.pm TARGET: home/httpd/lib/perl/Apache/lonmenu.pm	Has routines which control the remote control.
SOURCE: rat/lonpageflip.pm TARGET: home/httpd/lib/perl/Apache/lonpageflip.pm	Deals with forward, backward, and other page flips.
SOURCE: rat/lonratedt.pm TARGET: home/httpd/lib/perl/Apache/lonratedt.pm	Builds up frame set and loads in the right thing.
SOURCE: loncom/html/res/adm/pages/homeworkmenu.html TARGET: home/httpd/html/res/adm/pages/homeworkmenu.html	Homework remote control.
SOURCE: loncom/html/res/adm/pages/annotator/admannotations.pm TARGET: home/httpd/lib/perl/Apache/admannotations.pm	This will take annotations and then plug them into a page.

SOURCE: loncom/html/res/adm/pages/bookmarkmenu/admbookmarks.pm TARGET: home/httpd/lib/perl/Apache/admbookmarks.pm	This will take bookmarks and get/write/display them for the LON-CAPA user interface.
SOURCE: rat/lonratsrv.pm TARGET: home/httpd/lib/perl/Apache/lonratsrv.pm	Handler that takes output from RAT and stores it on disk. Handles the upper hidden frame of the added window that comes up in RAT. (3 frames come up in RAT server, code, and output. This module handles server connection.)
SOURCE: rat/lonpage.pm TARGET: home/httpd/lib/perl/Apache/lonpage.pm	bundles pages into one page
SOURCE: rat/lonuserstate.pm TARGET: home/httpd/lib/perl/Apache/lonuserstate.pm	compile course into binary data structure (in loncom/rat)
SOURCE: loncom/xml/lontex.pm TARGET: home/httpd/lib/perl/Apache/lontex.pm	Handler for tex files (somewhere in modules)
SOURCE: loncom/xml/lontexconvert.pm TARGET: home/httpd/lib/perl/Apache/lontexconvert.pm	Access to tth/ttm
SOURCE: loncom/xml/lonxml.pm TARGET: home/httpd/lib/perl/Apache/lonxml.pm	XML Parsing Module
SOURCE: loncom/xml/lonplot.pm TARGET: home/httpd/lib/perl/Apache/lonplot.pm	XML-based plotter of graphs
SOURCE: loncom/xml/style.pm TARGET: home/httpd/lib/perl/Apache/style.pm	Style Parsing Module
SOURCE: loncom/xml/londefdef.pm TARGET: home/httpd/lib/perl/Apache/londefdef.pm	Tags Default Definition Module
SOURCE: loncom/xml/run.pm TARGET: home/httpd/lib/perl/Apache/run.pm	used to prevent poorly written problems from causing lingering after effects
SOURCE: loncom/xml/scripttag.pm TARGET: home/httpd/lib/perl/Apache/scripttag.pm	implements <script>, <scriptlib>, <parserlib>, and <import>
SOURCE: loncom/homework/randomlabel.pm TARGET: home/httpd/lib/perl/Apache/randomlabel.pm	Interface for producing applet code which randomizes the labelling of an image.
SOURCE: loncom/homework/lonhomework.pm TARGET: home/httpd/lib/perl/Apache/lonhomework.pm	handles requests for output, evaluation, and alteration of a homework resource
SOURCE: loncom/homework/inputtags.pm TARGET: home/httpd/lib/perl/Apache/inputtags.pm	produces HTML input tags (<INPUT>) for rendering homework resources
SOURCE: loncom/homework/structuretags.pm TARGET: home/httpd/lib/perl/Apache/structuretags.pm	produces HTML tags necessary for structuring the presentation of homework resources

SOURCE: loncom/homework/response.pm TARGET: home/httpd/lib/perl/Apache/response.pm	defines different types of responses given to student as well as syntax for producing response values
SOURCE: loncom/homework/caparesponse/caparesponse.pm TARGET: home/httpd/lib/perl/Apache/caparesponse.pm	handles request to the CAPA homework processing engine
SOURCE: loncom/xml/Safe.pm TARGET: home/httpd/lib/perl/Safe.pm	Meant to safely substitute for CPAN version of Safe. Allows for safely executing embedded perl comands in a way that does not threaten the operating system.
SOURCE: loncom/auth/localauth.pm TARGET: home/httpd/lib/perl/localauth.pm	Local authentication mechanism (meant to be customized).
SOURCE: loncom/auth/lonacc.pm TARGET: home/httpd/lib/perl/Apache/lonacc.pm	(This module, like loncacc.pm also authenticates with cookies.) lonacc.pm coordinates access to a wide range of administrative-type functions (e.g. roles, logout, annotations, and bookmarks) as well as coordinating access to educational resources.
SOURCE: loncom/auth/lonracc.pm TARGET: home/httpd/lib/perl/Apache/lonracc.pm	access handler for file transfers
SOURCE: loncom/auth/loncacc.pm TARGET: home/httpd/lib/perl/Apache/loncacc.pm	This module provides access to an educational resource construction area. This module is invoked by the URL-related pattern syntax LocationMatch "^/priv.*" or LocationMatch "^/\~.*". Authentication of user identity is coordinated through cookies. The abbreviation "cacc" corresponds to "construction-space access"). If the cookie handle is invalid, then this module returns a forbidden status and makes appropriate log entries. If the cookie handle is valid, status is determined to be okay (and, for the "priv"-type access, the resource is delivered by the lonconstruct module).
SOURCE: loncom/auth/lonauth.pm TARGET: home/httpd/lib/perl/Apache/lonauth.pm	authenticate, set up session environment
SOURCE: loncom/auth/lonlogin.pm TARGET: home/httpd/lib/perl/Apache/lonlogin.pm	login screen
SOURCE: loncom/auth/lonlogout.pm TARGET: home/httpd/lib/perl/Apache/lonlogout.pm	logout
SOURCE: loncom/lonnet/perl/lonrep.pm TARGET: home/httpd/lib/perl/Apache/lonrep.pm	replication
SOURCE: loncom/auth/lonroles.pm TARGET: home/httpd/lib/perl/Apache/lonroles.pm	This perl handling module reads in the

	(different courses, different privileges, etc) and produces a form-element HTML page which allows the user to select which role he wishes to exercise in the LON-CAPA system. For instance, a user may want to select between being a student in a thermodynamics physics course or a teaching assistant for an introductory calculus class.
SOURCE: loncom/interface/lonindexer.pm TARGET: home/httpd/lib/perl/Apache/lonindexer.pm	cross server filesystem browser
SOURCE: loncom/interface/groupsort.pm TARGET: home/httpd/lib/perl/Apache/groupsort.pm	Implements a second phase of importing multiple resources into the RAT. Allows for reordering the sequence of resources.
SOURCE: loncom/lonnet/perl/lonnet.pm TARGET: home/httpd/lib/perl/Apache/lonnet.pm	This file is an interface to the lonc processes of the LON-CAPA network as well as set of elaborated functions for handling information necessary for navigating through a given cluster of LON-CAPA machines within a domain. There are over 40 specialized functions in this module which handle the reading and transmission of metadata, user information (ids, names, environments, roles, logs), file information (storage, reading, directories, extensions, replication, embedded styles and descriptors), educational resources (course descriptions, section names and numbers), url hashing (to assign roles on a url basis), and translating abbreviated symbols to and from more descriptive phrases or explanations.
SOURCE: loncom/homework/lectureonline.lcpm TARGET: home/httpd/html/res/adm/includes/lectureonline.lcpm	Intended for providing perl functions for the <script></script> environment in a problem that was converted from lectureonline.
SOURCE: loncom/homework/default_homework.lcpm TARGET: home/httpd/html/res/adm/includes/default_homework.lcpm	Provides many functions for the <script> environment in a .problem. Functions are documented in CVS:loncapa/doc/homework/homework5.html.

Session Three: Roles and lonnet/loncommon useful functions (Guy)

NAME

Apache::lonnet - Subroutines to ask questions about things in the network.

SYNOPSIS

Invoked by other LON-CAPA modules, when they need to talk to or about objects in the network.

```
&Apache::lonnet::SUBROUTINENAME(ARGUMENTS);
```

Common parameters:

- $uname : an internal username (if $cname expecting a course Id specifically)
- $udom : a domain (if $cdom expecting a course's domain specifically)
- $symb : a resource instance identifier
- $namespace : the name of a .db file that contains the data needed or being set.

INTRODUCTION

This module provides subroutines which interact with the lonc/lond (TCP) network layer of LON-CAPA. And Can be used to ask about - classes - users - resources

For many of these objects you can also use this to store data about them or modify them in various ways.

This is part of the LearningOnline Network with CAPA project described at http://www.lon-capa.org.

RETURN MESSAGES

- con_lost : unable to contact remote host
- con_delayed : unable to contact remote host, message will be delivered when the connection is brought back up
- con_failed : unable to contact remote host and unable to save message for later delivery
- error: : an error a occured, a description of the error follows the :
- no_such_host : unable to fund a host associated with the user/domain that was requested

---

Session Environment Functions

- `appenv(%hash)` : the value of %hash is written to the user envirnoment file, and will be restored for each access this user makes during this session, also modifies the %ENV for the current process
- `delenv($regexp)` : removes all items from the session environment file that matches the regular expression in $regexp. The values are also delted from the current processes %ENV.

User Information

- `queryauthenticate($uname,$udom)` : try to determine user's current authentication scheme
- `authenticate($uname,$upass,$udom)` : try to authenticate user from domain's lib servers (first use the current one), $upass should be the users password
- `homeserver($uname,$udom)` : find the server which has the user's directory and files (there must be only one) . This caches the answer and also caches if there is an error.
- `idget($udom,@ids)` : find the usernames behind a list of IDs (IDs are a unique resource in a domain, there must be only 1 ID per username, and only 1 username per ID in a specific domain) (returns hash: id=>name,id=>name)
- `idrget($udom,@unames)` : find the IDs behind a list of usernames (returns hash: name=>id,name=>id)
- `idput($udom,%ids)` : store away a list of names and associated IDs
- `rolesinit($udom,$username,$authhost)` : get user privileges
- `usection($udom,$uname,$cname)` : finds the section of student in the course $cname, return section name/number or '' for ``not in course'' and '-1' for ``no section''
- `userenvironment($udom,$uname,@what)` : gets the values of the keys passed in @what from the requested user's environment, returns a hash

User Roles

- `allowed($priv,$uri)` : check for a user privilege; returns codes for allowed actions F: full access U,I,K: authentication modes (cxx only) '': forbidden 1: user needs to choose course 2: browse allowed
- `definerole($rolename,$sysrole,$domrole,$courole)` : define role; define a custom role rolename set privileges in format of lonTabs/roles.tab for system, domain, and course level
- `plaintext($short)` : return value in %prp hash (rolesplain.tab); plain text explanation of a user role term

User Modification

- `assignrole($udom,$uname,$url,$role,$end,$start)` : assign role; give a role to a user for the level given by URL. Optional start and end dates (leave empty string or zero for ``no date'')

- `changepass($uname,$udom,$currentpass,$newpass,$server)` : attempts to change a users, password, possible return values are: `ok, pwchange_failure, non_authorized, auth_mode_error, unknown_user, refused`
- `modifyuserauth($udom,$uname,$umode,$upass)` : modify user authentication
- `modifyuser($udom,$uname,$uid,$umode,$upass,$first,$middle,$last,$gene)` : modify user
- modifystudent($udom,$uname,$uid,$umode,$upass,$first,$middle,$last,$gene,$usec, $end,$start) : modify student
- `assigncustomrole($udom,$uname,$url,$rdom,$rnam,$rolename,$end,$start)` : assign custom role; give a custom role to a user for the level given by URL. Specify name and domain of role author, and role name
- `revokerole($udom,$uname,$url,$role)` : revoke a role for url
- `revokecustomrole($udom,$uname,$url,$role)` : revoke a custom role

Course Infomation

- `coursedescription($courseid)` : course description
- `courseresdata($coursenum,$coursedomain,@which)` : request for current parameter setting for a specific course, @what should be a list of parameters to ask about. This routine caches answers for 5 minutes.

Course Modification

- `writecoursepref($courseid,%prefs)` : write preferences (environment database) for a course
- `createcourse($udom,$description,$url)` : make/modify course

Resource Subroutines

- `subscribe($fname)` : subscribe to a resource, returns URL if possible (probably should use repcopy instead)
- `repcopy($filename)` : subscribes to the requested file, and attempts to replicate from the owning library server, Might return HTTP_SERVICE_UNAVAILABLE, HTTP_NOT_FOUND, FORBIDDEN, OK, or HTTP_BAD_REQUEST, also attempts to grab the metadata for the resource. Expects the local filesystem pathname (/home/httpd/html/res/....)

Resource Information

- EXT($varname,$symb,$udom, $uname) : evaluates and returns the value of a vairety of different possible values, $varname should be a request string, and the other parameters can be used to specify who and what one is asking about.

  Possible values for $varname are environment.lastname (or other item from the environment hash), user.name (or someother aspect about the user), resource.0.maxtries (or some other part and parameter of a resource)

- `directcondval($number)` : get current value of a condition; reads from a state string
- `condval($condidx)` : value of condition index based on state
- `metadata($uri,$what,$liburi,$prefix,$depthcount)` : request a resource's metadata, $what should be either a specific key, or either 'keys' (to get a list of possible keys) or 'packages' to get a list of packages that this resource currently uses, the last 3 arguments are only used internally for recursive metadata. This function automatically caches all requests
- `metadata_query($query,$custom,$customshow)` : make a metadata query against the network of library servers; returns file handle of where SQL and regex results will be stored for query
- `symbread($filename)` : return symbolic list entry (filename argument optional); returns the data handle
- `symbverify($symb,$thisfn)` : verifies that $symb actually exists and is a possible symb for the URL in $thisfn, returns a 1 on success, 0 on failure, user must be in a course, as it assumes the existance of the course init hash, and uses $ENV('request.course.id'}
- `symbclean($symb)` : removes versions numbers from a symb, returns the cleaned symb
- `is_on_map($uri)` : checks if the $uri is somewhere on the current course map, user must be in a course for it to work
- `numval($salt)` : return random seed value (addend for rndseed)
- `rndseed($symb,$courseid,$udom,$username)` : create a random sum; returns a random seed, all arguments are optional, if they aren't sent it uses the environment to derive them. Note: if symb isn't sent and it can't get one from &symbread it will use the current time as its return value
- `ireceipt($funame,$fudom,$fucourseid,$fusymb)` : return unique, unfakeable, receipt
- `receipt()` : API to ireceipt working off of ENV values; given out to users
- `countacc($url)` : count the number of accesses to a given URL
- `checkout($symb,$tuname,$tudom,$tcrsid)` : creates a record of a user having looked at an item, most likely printed out or otherwise using a resource
- `checkin($token)` : updates that a resource has beeen returned (a hard copy version for instance) and returns the data that $token was Checkout with ($symb, $tuname, $tudom, and $tcrsid)
- `expirespread($uname,$udom,$stype,$usymb)` : set expire date for spreadsheet
- `devalidate($symb)` : devalidate temporary spreadsheet calculations, forcing spreadsheet to reevaluate the resource scores next time.

Storing/Retrieving Data

- `store($storehash,$symb,$namespace,$udom,$uname)` : stores hash permanently for this url; hashref needs to be given and should be a \%hashname; the remaining args aren't required and if they aren't passed or are '' they will be derived from the ENV

- `cstore($storehash,$symb,$namespace,$udom,$uname)` : same as store but uses critical subroutine
- `restore($symb,$namespace,$udom,$uname)` : returns hash for this symb; all args are optional
- `tmpstore($storehash,$symb,$namespace,$udom,$uname)` : storage that works very similar to store/cstore, but all data is stored in a temporary location and can be reset using tmpreset, $storehash should be a hash reference, returns nothing on success
- `tmprestore($symb,$namespace,$udom,$uname)` : storage that works very similar to restore, but all data is stored in a temporary location and can be reset using tmpreset. Returns a hash of values on success, error string otherwise.
- `tmpreset($symb,$namespace,$udom,$uname)` : temporary storage reset, deltes all keys for $symb form the temporary storage hash.
- `get($namespace,$storearr,$udom,$uname)` : returns hash with keys from array reference filled in from namesp ($udomain and $uname are optional)
- `del($namespace,$storearr,$udom,$uname)` : deletes keys out of array from namesp ($udomain and $uname are optional)
- `dump($namespace,$udom,$uname,$regexp)` : dumps the complete (or key matching regexp) namespace into a hash ($udomain, $uname and $regexp are optional)
- `put($namespace,$storehash,$udomain,$uname)` : stores hash in namesp ($udomain and $uname are optional)
- `cput($namespace,$storehash,$udomain,$uname)` : critical put ($udomain and $uname are optional)
- `eget($namespace,$storearr,$udomain,$uname)` : returns hash with keys from array reference filled in from namesp (encrypts the return communication) ($udomain and $uname are optional)
- `log($udom,$name,$home,$message)` : write to permanent log for user; use critical subroutine

Network Status Functions

- `dirlist($uri)` : return directory list based on URI
- `spareserver()` : find server with least workload from spare.tab

Apache Request

- `ssi($url,%hash)` : server side include, does a complete request cycle on url to localhost, posts hash

Data to String to Data

- `hash2str(%hash)` : convert a hash into a string complete with escaping and '=' and '&' separators, supports elements that are arrayrefs and hashrefs

- `hashref2str($hashref)` : convert a hashref into a string complete with escaping and '=' and '&' separators, supports elements that are arrayrefs and hashrefs
- `arrayref2str($arrayref)` : convert an arrayref into a string complete with escaping and '&' separators, supports elements that are arrayrefs and hashrefs
- `str2hash($string)` : convert string to hash using unescaping and splitting on '=' and '&', supports elements that are arrayrefs and hashrefs
- `str2array($string)` : convert string to hash using unescaping and splitting on '&', supports elements that are arrayrefs and hashrefs

Logging Routines

These routines allow one to make log messages in the lonnet.log and lonnet.perm logfiles.

- `logtouch()` : make sure the logfile, lonnet.log, exists
- `logthis()` : append message to the normal lonnet.log file, it gets preiodically rolled over and deleted.
- `logperm()` : append a permanent message to lonnet.perm.log, this log file never gets deleted by any automated portion of the system, only messages of critical importance should go in here.

General File Helper Routines

- `getfile($file)` : serves up a file, returns the contents of a file or -1; replicates and subscribes to the file
- `filelocation($dir,$file)` : returns file system location of a file based on URI; meant to be ``fairly clean'' absolute reference, $dir is a directory that relative $file lookups are to looked in ($dir of /a/dir and a file of ../bob will become /a/bob)
- `hreflocation($dir,$file)` : returns file system location or a URL; same as filelocation except for hrefs
- `declutter()` : declutters URLs (remove docroot, beginning slashes, 'res' etc)

HTTP Helper Routines

- `escape()` : unpack non-word characters into CGI-compatible hex codes
- `unescape()` : pack CGI-compatible hex codes into actual non-word ASCII character

Private: Underlying communication routines (Shouldn't call)

- `subreply()` : tries to pass a message to lonc, returns con_lost if incapable
- `reply()` : uses subreply to send a message to remote machine, logs all failures
- `critical()` : passes a critical message to another server; if cannot get through then place message in connection buffer directory and returns con_delayed, if incapable of saving message, returns con_failed

- `reconlonc()` : tries to reconnect lonc client processes.

Private: Resource Access Logging

- `flushcourselogs()` : flush (save) buffer logs and access logs
- `courselog($what)` : save message for course in hash
- `courseacclog($what)` : save message for course using &courselog(). Perform special processing for specific resource types (problems, exams, quizzes, etc).
- `goodbye()` : flush course logs and log shutting down; it is called in srm.conf as a PerlChildExitHandler

Private: Other

`symblist($mapname,%newhash)` : update symbolic storage links

Coding Guidelines

Things to keep in mind while coding handlers for LON-CAPA

1. DON'T write to Access machine disks with permanent data, use store/restore
2. DON'T `use print()`, use `$request->print()`
3. DON'T launch children
4. DO use `$Apache::lonnet::perlvar{'lonDaemons'}/tmp` for temporary data.
5. DO query the return value of every file operation.
6. DO `use strict;`
7. DO `use strict;`
8. DO familiarize your self with the functions in `lonnet.pm` and use them to communicate with other servers and when you handler needs to ask questions.
9. DON'T use `&Apache::lonnet::reply()`

Session Four: How to install, Proper Debug Cycle for Handlers, Coding/Documentation Guidelines (Guy)

lpml.dtd

```
<!--
       Linux Packaging Markup Language version 1.0.

       Date: May 3, 2001
       Author: Scott Harrison

       Linux Packaging Markup Language aims to capture
       a source code development tree and effectively map
       that to a compilation and installation process for
       generating software packages and updating a run-time
       system.  This language helps handle dependency information,
       file globbing, file permissions, file ownerships,
       different targets for different linux distributions,
       preservation of configuration information, directory
       permissions, directory ownerships, compilation commands,
       and, ultimately, system software status reports.
-->
<!--

Copyright Michigan State University Board of Trustees

This file is part of the LearningOnline Network with CAPA (LON-CAPA).

LON-CAPA is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

LON-CAPA is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with LON-CAPA; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

/home/httpd/html/adm/gpl.txt

http://www.lon-capa.org/

-->

<!ENTITY % lpml.Version
        "-//TUX//DTD lpml 1.0 Final//EN"
        >
<!-- Typical usage: -->
<!-- <!DOCTYPE lpml PUBLIC "-//TUX/DTD LPML 1.0 Final//EN"
      "http://lpml.sourceforge.net/DTD/lpml.dtd"> -->
<!-- <lpml> -->
<!-- ... -->
<!-- </lpml> -->

<!ELEMENT categoryname (#PCDATA)>
<!ELEMENT targetroot (#PCDATA)>
<!ELEMENT sourceroot (#PCDATA)>

<!ELEMENT target (#PCDATA)>
<!ATTLIST target
    dist CDATA #REQUIRED
    >
```

```
<!ELEMENT source (#PCDATA)>
<!ELEMENT targetdir (#PCDATA)>
<!ATTLIST targetdir
    dist CDATA #REQUIRED
    >
<!ELEMENT sourcedir (#PCDATA)>
<!ELEMENT glob (#PCDATA)>
<!ELEMENT build (#PCDATA)>
<!ELEMENT buildlink (#PCDATA)>

<!--
The trigger attribute of the build element is meant
to only have one of two possible values:
  "always run", or
  "run if dependencies change"

Note that you must type this text exactly in for the
attribute value to be understood and processed correctly
by the lpml "make build" parser.
-->
<!ELEMENT lpml
 (targetroot|sourceroot|specialnotices|files|categories|directories|rpm)+>
<!ATTLIST build
   trigger CDATA #REQUIRED
>
<!ELEMENT specialnotices (specialnotice)+>
<!ELEMENT categories (category)+>
<!ELEMENT directories (directory)+>
<!ELEMENT files (file|fileglob|link)+>
<!ELEMENT dependencies (#PCDATA)>
<!ELEMENT note (#PCDATA|table|b|br)*>
<!ELEMENT b (#PCDATA)>

<!ELEMENT specialnotice (#PCDATA)>
<!ATTLIST specialnotice
    dist CDATA #REQUIRED
    >
<!ELEMENT category (chmod,chown,abbreviation)>
<!ATTLIST category
    type CDATA #REQUIRED
    name CDATA #REQUIRED
    >
<!ELEMENT chown (#PCDATA)>
<!ATTLIST chown
    dist CDATA #REQUIRED
    >
<!ELEMENT chmod (#PCDATA)>
<!ATTLIST chmod
    dist CDATA #REQUIRED
    >

<!ELEMENT abbreviation (#PCDATA)>
<!ELEMENT br EMPTY>
<!ELEMENT nobr EMPTY>
<!ELEMENT table (#PCDATA|tr)*>
<!ATTLIST table
    cellpadding CDATA #IMPLIED
    cellspacing CDATA #IMPLIED
    border CDATA #IMPLIED
    >
<!ELEMENT tr (#PCDATA|td)*>
<!ELEMENT td (#PCDATA|br|ul|tt|nobr)*>
<!ELEMENT description (#PCDATA|br|tt|u)*>
<!ELEMENT ul (#PCDATA|li)*>
<!ELEMENT li (#PCDATA)>
<!ELEMENT tt (#PCDATA)>
<!ELEMENT u (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT linkto (#PCDATA)>

<!ELEMENT directory (targetdir+,categoryname,description?)>
```

```
<!ATTLIST directory
    dist CDATA #REQUIRED
    >

<!ELEMENT file (source,target+,categoryname,description?,buildlink?,
                note?,build?,status?,dependencies?)>
<!ELEMENT link (linkto,target,categoryname,description?,
                note?,build?,status?,dependencies?)>

<!ELEMENT fileglob (glob,sourcedir,targetdir,categoryname,

description?,note?,build?,buildlink?,status?,dependencies?,filenames?)>
<!ELEMENT filenames (#PCDATA)>

<!ELEMENT rpm (rpmSummary,rpmName,rpmVersion,rpmRelease,rpmVendor,
               rpmBuildRoot,rpmCopyright,rpmGroup,rpmSource,rpmAutoReqProv,
               rpmdescription,rpmpre,rpmRequires)>
<!ELEMENT rpmSummary (#PCDATA)>
<!ELEMENT rpmName (#PCDATA)>
<!ELEMENT rpmVersion (#PCDATA)>
<!ELEMENT rpmRelease (#PCDATA)>
<!ELEMENT rpmVendor (#PCDATA)>
<!ELEMENT rpmBuildRoot (#PCDATA)>
<!ELEMENT rpmCopyright (#PCDATA)>
<!ELEMENT rpmGroup (#PCDATA)>
<!ELEMENT rpmSource (#PCDATA)>
<!ELEMENT rpmAutoReqProv (#PCDATA)>
<!ELEMENT rpmdescription (#PCDATA)>
<!ELEMENT rpmpre (#PCDATA)>
<!ELEMENT rpmRequires (item)+>
<!ELEMENT item (#PCDATA)>
```

piml.dtd

```
<!--
       Post Installation Markup Language version 1.0.

       Date: January 24, 2002
       Author: Scott Harrison

       Post Installation Markup Language works to perform
       'intelligent' modifications of existing files on
       a software system.  This allows for the reconfiguring
       of existing configuration files without having the
       overlap of files between software packages.  PIML
       also allows you to have file permission/ownership
       specifications different than that specified by the
       original software package.

       For instance, Apache web server configuration files
       are a popular target for many different software applications.
       A preferred approach is to simply append an 'Include ...'
       line to the Apache web server configuration files.

       Post Installation Markup Language is meant to be
       a natural wrapper to target installations generated
       by LPML (Linux Packaging Markup Language).

       Usages of Post Installation Markup Language are:
       * immediately after installation with LPML
       * generating '%post' syntax for RPMs
       * generating post-installation syntax for Debian packages.

       Dependency checking is supported as a poor man's probing....
       There is no point in reconfiguring a file that isn't installed,
       or is missing a system component for successful operation.

       Another neat use of PIML is to allow for invocation of various
       processes after installation.
-->
<!--

Copyright Michigan State University Board of Trustees

This file is part of the LearningOnline Network with CAPA (LON-CAPA).

LON-CAPA is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

LON-CAPA is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with LON-CAPA; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

/home/httpd/html/adm/gpl.txt

http://www.lon-capa.org/

-->

<!ENTITY % piml.Version
        "-//TUX//DTD piml 1.0 Final//EN">

<!-- Typical usage:

            <!DOCTYPE piml PUBLIC "-//TUX//DTD PIML 1.0 Final//EN"
            "http://lpml.sourceforge.net/DTD/piml.dtd">
            <piml>

            </piml>
 -->

<!ELEMENT categoryname (#PCDATA)>
<!ELEMENT targetroot (#PCDATA)>

<!ELEMENT target (#PCDATA)>
<!ATTLIST target
    dist CDATA #REQUIRED
    >
<!ELEMENT piml (targetroot|specialnotices|files|categories)+>
<!ELEMENT specialnotices (specialnotice)+>
<!ELEMENT categories (category)+>
<!ELEMENT files (file)+>
<!ELEMENT dependencies (#PCDATA)>
<!ATTLIST dependencies
    dist CDATA #REQUIRED
    >
<!ELEMENT note (#PCDATA)>

<!ELEMENT specialnotice (#PCDATA)>
<!ELEMENT category (chmod,chown,abbreviation)>
<!ATTLIST category
    type CDATA #REQUIRED
    name CDATA #REQUIRED
    >
<!ELEMENT chown (#PCDATA)>
<!ATTLIST chown
    dist CDATA #REQUIRED
    >
<!ELEMENT chmod (#PCDATA)>
<!ATTLIST chmod
    dist CDATA #REQUIRED
    >
```

```
<!ELEMENT abbreviation (#PCDATA)>

<!ELEMENT file (target,categoryname?,
                note?,status?,dependencies?,perlscript)>
<!ELEMENT TARGET EMPTY>
<!ELEMENT perlscript (#PCDATA|TARGET)*>
<!ATTLIST perlscript
    mode CDATA #REQUIRED
    >
```

xfml.dtd

```
<!--
       XML Filter Markup Language version 1.0.

       Date: January 27, 2002
       Author: Scott Harrison

       XML Filter Markup Language works to extract those sections
       of an XML document matching certain conditions.  This, in some
       respects, relies on functionality similar to that expected
       with XSL-type files.  (I do not use standard XSL syntax because
       I am not convinced that it would be simple and elegant for
       this particular task).

       This is anticipated to be a very important feature associated
       with a larger Makefile-ish type approach involving the XML dtds:
       LPML and PIML.
       LPML (Linux Packaging Markup Language) can work to install files
       on a single computer or a distributed network of computers with
       pre-configured ssh accessibility (.ssh/authorized_keys, etc).
       PIML (Post Installation Markup Language) works to coordinate the
       last little scripts that need to be run after an installation.
       With all this make-"power", wouldn't it be nice to only pass through
       portions of an LPML specification?

-->
<!--

Copyright Michigan State University Board of Trustees

This file is part of the LearningOnline Network with CAPA (LON-CAPA).

LON-CAPA is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

LON-CAPA is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with LON-CAPA; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA

/home/httpd/html/adm/gpl.txt

http://www.lon-capa.org/

-->

<!-- Note: current status is that choice:include is currently unsupported and
     all matching statements REQUIRE two levels of 'when' statements;
     not one or more than two...  okay.. this will be fixed soon -->
```

```
<!ENTITY % xfml.Version
        "-//TUX//DTD piml 1.0 Final//EN">

<!-- Typical usage:

            <!DOCTYPE xfml PUBLIC "-//TUX//DTD PIML 1.0 Final//EN">
            <xfml>

            </xfml>
-->

<!ELEMENT choice:include (#PCDATA)>
<!ELEMENT choice:exclude (#PCDATA)>

<!ELEMENT xfml (clause)+>
<!ELEMENT clause (when:cdata|when:name|when:attribute|choice:exclude)+>
<!ELEMENT when:name
 (when:cdata|when:name|when:attribute|choice:include|choice:exclude)+>
<!ELEMENT when:attribute
  (when:cdata|when:name|when:attribute|choice:include|
  choice:exclude)+>
<!ELEMENT when:cdata
  (when:cdata|when:name|when:attribute|choice:include|
  choice:exclude)+>

<!ATTLIST when:name
    match CDATA #REQUIRED>
<!ATTLIST when:attribute
    match CDATA #REQUIRED>
<!ATTLIST when:cdata
    match CDATA #REQUIRED>
<!ATTLIST choice:include
    nodename CDATA #REQUIRED>
<!ATTLIST choice:exclude
    nodename CDATA #REQUIRED>
```

Session Five: Worktime (make new handler, debugging/testing) (Guy)

Day 2

Session One: Roles, Data Storage, Parameters (Gerd)

Domains

Every user in LON-CAPA is member of one domain. A domain can be institutional and "open", for example "msu" or "wscc" - open means that in it there can be students, authors and other users. A domain can also be functional, for example "timss_tests" or "smith_publishers". Physically, every domain needs at least one dedicated library server.

Userdata

Every user in the system has one library server, which is their home server. It stores the authoritative copy of all of their records. Internally, this data is stored in a directory

`/home/httpd/lonUsers/`*`domain`*`/`*`1.char`*`/`*`2.char`*`/`*`3.char`*`/`*`username`*`/`

for example

`/home/httpd/lonUsers/msu/s/m/i/smith/`

```
ls -alF /home/httpd/lonUsers/msu/k/o/r/kortemey

-rw-r--r--   1 www      users       13006 May 15 12:21 activity.log
-rw-r-----   1 www      users       12413 Oct 26  2000 coursedescriptions.db
-rw-r--r--   1 www      users       11361 Oct 26  2000 coursedescriptions.hist
-rw-r-----   1 www      users       13576 Apr 19 17:45 critical.db
-rw-r--r--   1 www      users        1302 Apr 19 17:45 critical.hist
-rw-r-----   1 www      users       13512 Apr 19 17:45 email_status.db
-rw-r--r--   1 www      users        1496 Apr 19 17:45 email_status.hist
-rw-r--r--   1 www      users       12373 Apr 19 17:45 environment.db
-rw-r--r--   1 www      users         169 Apr 19 17:45 environment.hist
-rw-r-----   1 www      users       12315 Oct 25  2000 junk.db
-rw-r--r--   1 www      users        1590 Nov  4  1999 junk.hist
-rw-r-----   1 www      users       23626 Apr 19 17:45 msu_12679c3ed543a25msul1.db
-rw-r--r--   1 www      users        3363 Apr 19 17:45 msu_12679c3ed543a25msul1.hist
-rw-r-----   1 www      users       17242 Nov 13  2000 msu_1827338c7d339a3msul1.db
-rw-r--r--   1 www      users        1986 Nov 13  2000 msu_1827338c7d339a3msul1.hist
-rw-r-----   1 www      users       18497 Dec 21 11:25 msu_1827338c7d339b4msul1.db
-rw-r--r--   1 www      users        3801 Dec 21 11:25 msu_1827338c7d339b4msul1.hist
-rw-r-----   1 www      users       12470 Apr 19 17:45 nohist_annotations.db
-rw-r-----   1 www      users       13395 Nov 15  2000 nohist_bookmarks.db
-rw-r-----   1 www      users      104264 Apr 19 17:45
                              nohist_calculatedsheets_msu_12679c3ed543a25msul1.db
-rw-r-----   1 www      users       13248 Apr  5 17:18
                              nohist_calculatedsheets_msu_1827338c7d339b4msul1.db
-rw-r-----   1 www      users       12568 Oct 28  2000 nohist_coursedescriptions.db
-rw-r-----   1 www      users      765954 Apr 19 17:45 nohist_email.db
-rw-r--r--   1 www      users      710631 Apr 19 17:45 nohist_email.hist
-rw-r--r--   1 www      users          13 Apr 19 17:45 passwd
-rw-r--r--   1 www      users       12802 May  3 13:08 roles.db
-rw-r--r--   1 www      users        1316 Apr 12 16:05 roles.hist
```

Fig.2.1.1 – Directory listing of user's home directory

Files ending on `.db` are GDBM files, files ending on `.hist` are logs of entries to these files. Filenames starting with "nohist" do not keep history files. `passwd` stores the login mechanism and password (if applicable).

`environment` stores name-value pairs that are automatically added to the session environment at login time, for example the full name, etc.

`roles` stores the userroles.

`critical`, `nohist_email`, and `email_status` are used by the messaging mechanisms

Files with a course-ID as name, for example `msu_12679c3ed543a25msul1.db`, store performance data for that student in the course, as stored by `store` and `restore` in lonnet.

Other files are caches, for example for previously calculated spreadsheets, etc.

Courses

Courses are assigned to users, not vice versa. Internally, courses are handled like users without login privileges. The username is a unique ID, for example `msu_12679c3ed543a25msul1` – every course in every semester has a unique ID, there is no semester transition. The userdata of the course includes the full name of the course, a pointer to its top-level resource map ("course map"), and any associated deadlines, spreadsheets, etc., as well as a course enrollment list. The latter is somewhat redundant, since in principle, this list could be produced by going through the roles of all users, and looking for the valid role of being student in that course.

```
ls -alF /home/httpd/lonUsers/msu/1/2/6/12679c3ed543a25msul1/

-rw-r-----   1 www      users       17155 Apr 25 16:20 classlist.db
-rw-r--r--   1 www      users       60912 Apr 25 16:20 classlist.hist
-rw-r-----   1 www      users       12354 Jan  4 16:40 environment.db
-rw-r--r--   1 www      users          82 Jan  4 16:40 environment.hist
-rw-r-----   1 www      users      103030 May 15 14:47 nohist_calculatedsheets.db
-rw-r-----   1 www      users       13050 May  9 21:04 nohist_expirationdates.db
-rw-r--r--   1 www      users           6 Jan  4 16:40 passwd
-rw-r-----   1 www      users       17457 May  9 21:04 resourcedata.db
-rw-r--r--   1 www      users        8888 May  9 21:04 resourcedata.hist
```

Fig.2.1.2 – Directory listing of course's home directory

`classlist` is this list of students in the course, `environment` includes the course's full name, etc, and `resourcedata` are deadlines, etc (parameters for homework).

Roles

Users keep their login, data, preferences, etc, over their complete tenure. Every user can have several roles, and the roles can change over the lifetime of a username. For example, over the course of studies, a student username assumes the role of "student" in different courses. Roles can have start and expiration dates.

Example: User smith at msu		
Instructor	`msu_12679c3ed543a25msul1`	
Course Coordinator	`msu_12679c3ed543a25msul1`	From July 1st, 2001 to December 30th, 2001
Instructor	`msu_18879c3ed543a25msul2`	From Jan 1st, 2001 to June 30th, 2001
Resource Author	`msu`	From Aug 15th, 2000
Student	`msu_82679c3gd543a35msul1`	From July 1st, 2001 to December 30th, 2001

Fig.2.1.3 – Sample Instructor Roles

Example: User jones at msu		
Custom Role "Helproom TA (smith at msu)"	`msu_82679c3gd543a35msul1`	From July 1st, 2001 to December 30th, 2001
Student	`msu_02679c3gq543a35msul1`	From Jan 1st, 2001 to June 30th, 2001
Student	`umn_82679c3gd543a35umnl2`	From July 1st, 2001 to December 30th, 2001
Exam Proctor	`msu_82679c3gd543a35msul1`	Feb 21st, 2001, 1pm to 3pm

Fig.2.1.4 – Sample Student Roles

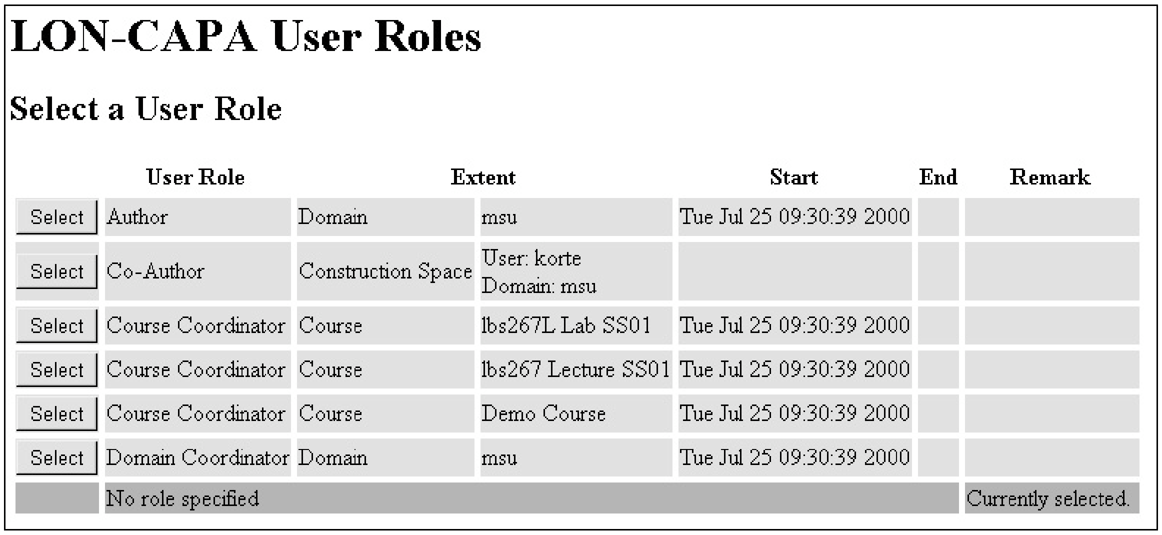
Custom Roles

Course Coordinators are able to define named "Custom Roles" for their courses within a pre-defined set of capabilities. In addition to these custom roles, there are three standard course faculty/staff roles defined, Instructor, Exam Proctor and TA. The instructor of record in a small class is likely to be "Course Coordinator" and "Instructor" during the term when the course is running, and might remain course coordinator afterwards. Course coordinator can assign themselves new roles for their course anytime.

Custom role definitions are stored in the `roles.db` file of the role author.

Choose a Role, Role Privileges

`lonroles` is a handler that allows a user to switch roles in mid-session. LON-CAPA attempts to work with "No Role Specified" as widely as possible, but certain handlers for example need specification which course they should act on, etc. Both in this scenario, and when the handler determines via `lonnet`'s `&allowed` function that a certain action is not allowed, `lonroles` is used as errorhandler. `lonroles` can also be accessed via the CRS button in the Remote Control. **Fig. 2.1.5** shows a sample output of `lonroles`.



LON-CAPA User Roles

Select a User Role

	User Role	Extent		Start	End	Remark
Select	Author	Domain	msu	Tue Jul 25 09:30:39 2000		
Select	Co-Author	Construction Space	User: korte Domain: msu			
Select	Course Coordinator	Course	lbs267L Lab SS01	Tue Jul 25 09:30:39 2000		
Select	Course Coordinator	Course	lbs267 Lecture SS01	Tue Jul 25 09:30:39 2000		
Select	Course Coordinator	Course	Demo Course	Tue Jul 25 09:30:39 2000		
Select	Domain Coordinator	Domain	msu	Tue Jul 25 09:30:39 2000		
	No role specified					Currently selected.

Fig. 2.1.5 – Sample Roles Choice in `lonroles.pm`

System: /

- Browse resources
- Generate anonymous statistics
- Create a Course Custom Role
- Send internal email

Domain: msu

- Assemble resources
- Browse resources
- Grant/revoke role of Administrator (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Author (UNIX authenticated)
- Grant/revoke role of Co-Author (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Course Coordinator (UNIX authenticated)
- Grant/revoke Course Custom Role (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Domain Guest (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Exam Proctor (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Instructor (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Librarian (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Copy resources
- Grant/revoke role of Student (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Teaching Assistant (UNIX authenticated, Internally authenticated, Kerberos authenticated)
- Create, edit, modify and publish resources
- Modify authentication mechanism and data for a user

Course: lbs267L Lab SS01

- Assemble resources
- Grant/revoke Course Custom Role (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Exam Proctor (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Instructor (Internally authenticated, Kerberos authenticated)
- Copy resources
- Grant/revoke role of Student (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Teaching Assistant (Internally authenticated, Kerberos authenticated)
- Create, edit, modify and publish resources
- Generate anonymous statistics
- Set assessment parameters
- Send broadcast and receipt-required email
- View grades

Course: lbs267 Lecture SS01

- Assemble resources
- Grant/revoke Course Custom Role (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Exam Proctor (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Instructor (Internally authenticated, Kerberos authenticated)
- Copy resources
- Grant/revoke role of Student (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Teaching Assistant (Internally authenticated, Kerberos authenticated)
- Create, edit, modify and publish resources
- Generate anonymous statistics
- Set assessment parameters
- Send broadcast and receipt-required email
- View grades

Course: Demo Course

- Assemble resources
- Grant/revoke Course Custom Role (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Exam Proctor (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Instructor (Internally authenticated, Kerberos authenticated)
- Copy resources
- Grant/revoke role of Student (Internally authenticated, Kerberos authenticated)
- Grant/revoke role of Teaching Assistant (Internally authenticated, Kerberos authenticated)
- Create, edit, modify and publish resources
- Generate anonymous statistics
- Set assessment parameters
- Send broadcast and receipt-required email
- View grades

Construction Space: User: korte, Domain: msu

Fig. 2.1.6 – Sample Set of Privileges

Fig. 2.1.6 shows a common set of privileges for the user roles in **Fig. 2.1.5**. The plain text explanations of the various roles and the extent of them is drawn from `/home/httpd/rolesplain.tab`, see **Fig. 2.1.7**.

```
[www@zaphod www]$ more /home/httpd/lonTabs/rolesplain.tab
```

```
s:system wide
d:domain wide
c:course wide
U:UNIX authenticated
I:Internally authenticated
K:Kerberos authenticated
C:according to course preferences
S:according to custom role settings
R:according to resource settings
L:unless locked
X:according to user session state
F:no restrictions
cm:No Role, Cumulative Privileges
su:Superuser
dc:Domain Coordinator
cc:Course Coordinator
in:Instructor
ta:Teaching Assistant
ep:Exam Proctor
cr:Course Custom Role
st:Student
ad:Administrator
li:Librarian
au:Author
dg:Domain Guest
ca:Co-Author
csu:Grant/revoke role of Superuser
cdc:Grant/revoke role of Domain Coordinator
ccc:Grant/revoke role of Course Coordinator
cin:Grant/revoke role of Instructor
cta:Grant/revoke role of Teaching Assistant
cep:Grant/revoke role of Exam Proctor
ccr:Grant/revoke Course Custom Role
cst:Grant/revoke role of Student
cad:Grant/revoke role of Administrator
cli:Grant/revoke role of Librarian
cau:Grant/revoke role of Author
cdg:Grant/revoke role of Domain Guest
cca:Grant/revoke role of Co-Author
mcr:Create a Course Custom Role
mau:Modify authentication mechanism and data for a user
bre:Browse resources
are:Assemble resources
cre:Copy resources
ere:Create, edit, modify and publish resources
mme:Modify metadata for a resource
vgr:View grades
mgr:Modify grades
gan:Generate anonymous statistics
dcm:Disable all communication among students
sma:Send internal email
srm:Send broadcast and receipt-required email
pch:Post to chatrooms and bulletin boards
dch:Delete messages from bulletin boards
pac:Post anonymously
rin:Get identity behind anonymous postings
las:Lock and unlock assessments
opa:Set assessment parameters
ain:Assume a student's identity
```

Fig. 2.1.7 – Explanation of Privilege Shorthands

Role Initialization

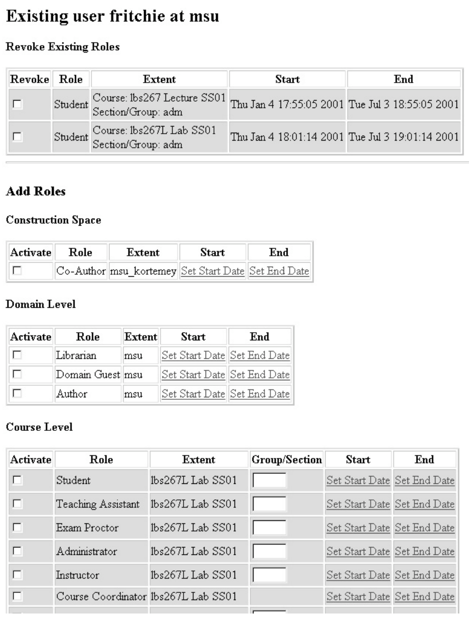
The privileges for a user are established at login time and stored in the session environment. A consequence is that a new role does not become active till the next login. Handlers are able to query for privileges using `lonnet`'s `&allowed` function. When a user first logs in, their role is the "common" role, which means that they have the sum of all of their privileges. During a session it might become necessary to choose a particular role, which as a consequence also limits the user to only the privileges in that particular role.

```
[www@zaphod www]$ more /home/httpd/lonTabs/roles.tab
su:s csu&U:sma:mau:cdc&U
dc:s sma
```

```
dc:d cli&UIK:cau&U:cdg&UIK:mau:ccc&U:cin&UIK:cta&UIK:cep&UIK:ccr&UIK:cst&UIK:cad&UIK
cc:s bre:sma:mcr
cc:c cin&IK:cta&IK:cep&IK:ccr&IK:cst&IK:are:cre:ere:vgr:gan:srm:opa
in:s sma
in:d bre
in:c vgr:mgr:gan:dcm:srm:pch:dch:pac:rin:las:opa
ta:d sma
ta:c bre&RL:vgr&CR:mgr&CR:srm:pch:dch:pac
ep:d sma
ep:c bre&R:mgr&R:dcm:las
cr:d sma
cr:c bre&R:vgr&SCR:mgr&SCR:gan&SCR:dcm&SC:srm&SC:pch:dch&S:pac:rin&S:las&SR:opa&SR
st:d sma&L
st:c bre&RXL:pch&L:pac&CL
ad:d sma
ad:c bre:gan:vgr:srm
li:s gan:sma
li:d mme
au:s gan:sma
au:d bre:are:cre:ere:cca&IK
ca:s gan:sma
ca:d bre:are:cre:ere
dg:d bre&R
```

Fig. 2.1.8 – Privileges by roles and extent

Role Assignment



Existing user fritchie at msu

Revoke Existing Roles

Revoke	Role	Extent	Start	End
☐	Student	Course: lbs267 Lecture SS01 Section/Group: adm	Thu Jan 4 17:55:05 2001	Tue Jul 3 18:55:05 2001
☐	Student	Course: lbs267L Lab SS01 Section/Group: adm	Thu Jan 4 18:01:14 2001	Tue Jul 3 19:01:14 2001

Add Roles

Construction Space

Activate	Role	Extent	Start	End
☐	Co-Author	msu_kortemey	Set Start Date	Set End Date

Domain Level

Activate	Role	Extent	Start	End
☐	Librarian	msu	Set Start Date	Set End Date
☐	Domain Guest	msu	Set Start Date	Set End Date
☐	Author	msu	Set Start Date	Set End Date

Course Level

Activate	Role	Extent	Group/Section	Start	End
☐	Student	lbs267L Lab SS01		Set Start Date	Set End Date
☐	Teaching Assistant	lbs267L Lab SS01		Set Start Date	Set End Date
☐	Exam Proctor	lbs267L Lab SS01		Set Start Date	Set End Date
☐	Administrator	lbs267L Lab SS01		Set Start Date	Set End Date
☐	Instructor	lbs267L Lab SS01		Set Start Date	Set End Date
☐	Course Coordinator	lbs267L Lab SS01		Set Start Date	Set End Date

Fig. 2.1.9 – Assigning privileges to a user

- `loncreateuser.pm` allows users to within their own privileges ('c*xx* privileges) create users and give them roles (**Fig. 2.1.9**)
- `londropadd.pm` allows course coordinators to upload courselists in different formats, and automatically create users (if they do not exist already), assign them the role of student in a course, and add them to the classlist.
- `loncreatecourse.pm` allows domain coordinators to create new courses and assign course coordinators.

Session Two: Spreadsheet and Messaging (Matthew)

Spreadsheets

The spreadsheet presents data on student performance on homework problems. Spreadsheets are handled by loncom/interface/lonspreadsheet.pm and are completely web based. A person who has selected a student role will access the spreadsheets using the [GRDS] button. A course coordinator is given access via the [SPRS] button. Students are not able to see data on anyone's performance but their own. Students are also not allowed to save spreadsheets.

Spreadsheet Structure and Hierarchy

The spreadsheets are laid out in the typical fashion, with some limitations. There can only be 52 columns, addressed [A-Za-z]. There may be any number of rows, but currently there do not exist facilities to add rows other than those automatically generated.

There are three levels of spreadsheets, as illustrated in **Fig. 2.2.1**.

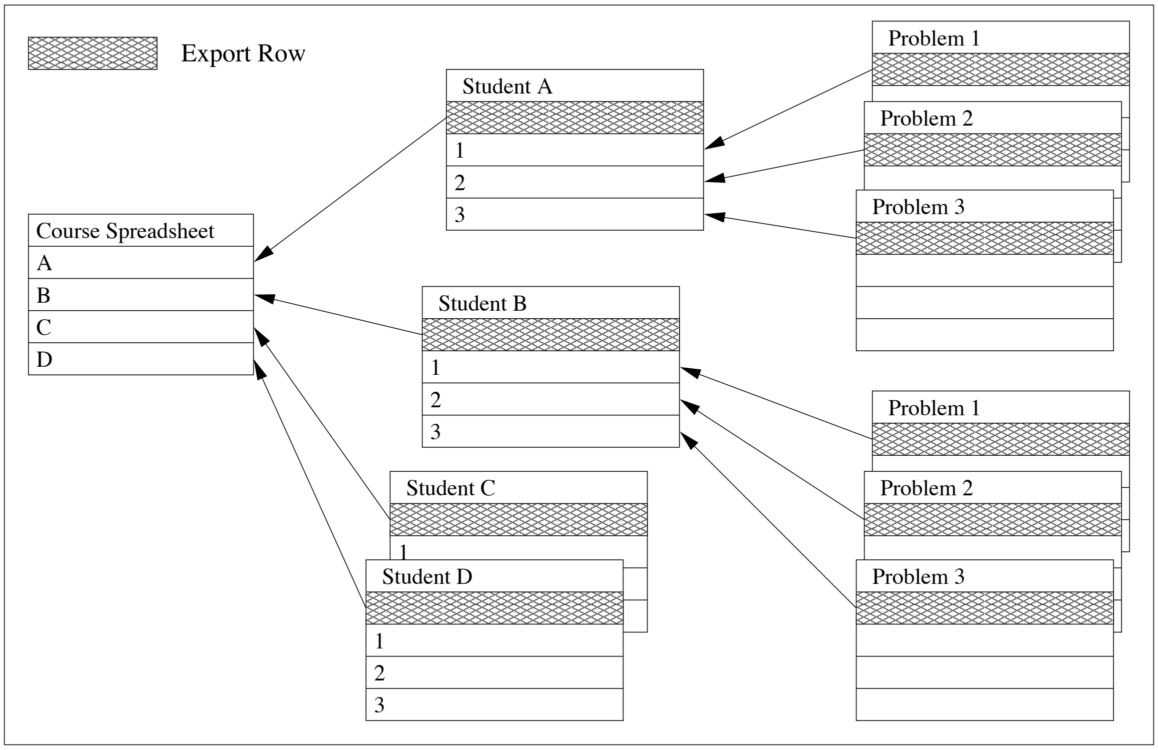


Fig. 2.1.1 – Spreadsheet Hierarchy

The rightmost spreadsheets are the assessment spreadsheets. The middle spreadsheets are the student spreadsheets. The left spreadsheet is the course spreadsheet.

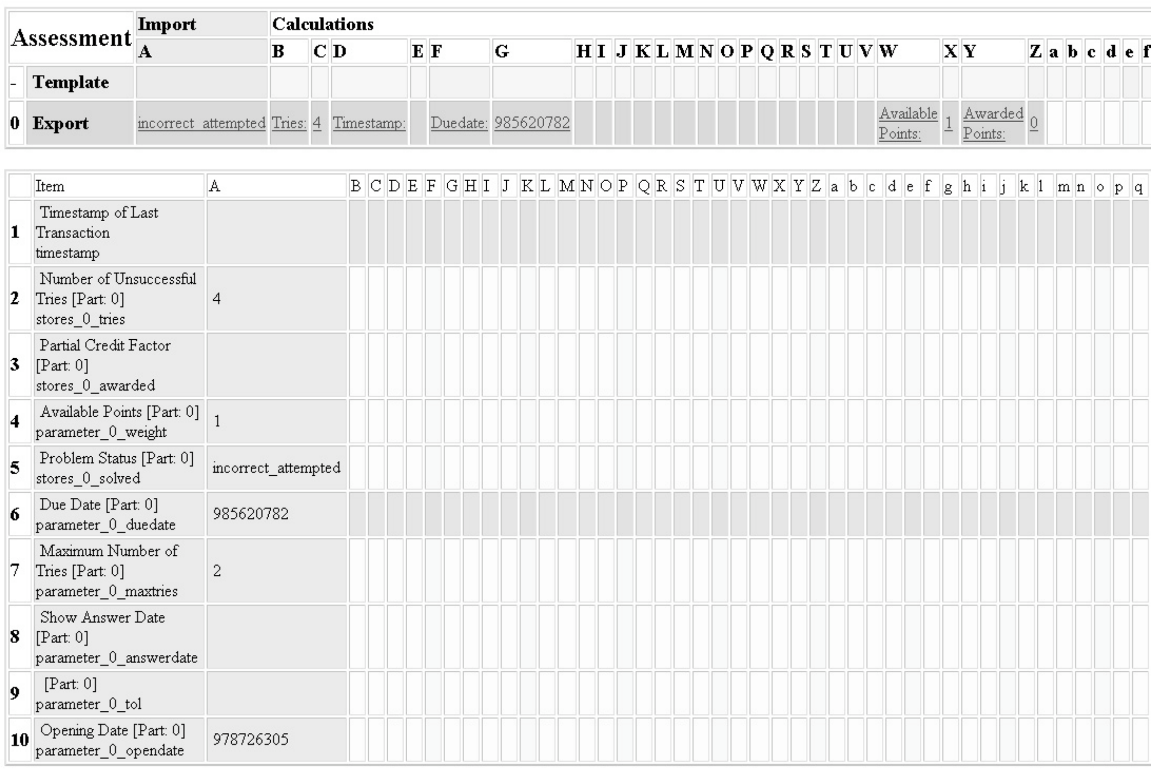
Export Rows

The hierarchy of spreadsheets described above allows data from the lower level spreadsheets (assessment and student) to be exported up to the higher level spreadsheets (student and course, respectively). Row 0 is the export row. Only the cells A0-Z0 are exported. Cells a0-z0 are not exported and can be used as 'scratch' space for the results exported in A0-Z0.

The export rows in the image are shown shaded in figure 2.2.1. Where the exported rows appear in the student and course spreadsheets is indicated by arrows.

Assessment Spreadsheet

The assessment spreadsheet gives data on the students performance on a specific resource in LON-CAPA (typically a *.problem resource). Parameters such as the due date, the number of tries possible, the number of attempts made, the correctness of the student solution, and any <parameter> tags inserted in the resource will be shown. **Fig. 2.2.2** shows an example of an assessment spreadsheet.



	Assessment	Import	Calculations																														
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f
-	Template																																
0	Export	incorrect_attempted	Tries:	4	Timestamp:		Duedate:	985620782																Available Points:	1	Awarded Points:	0						

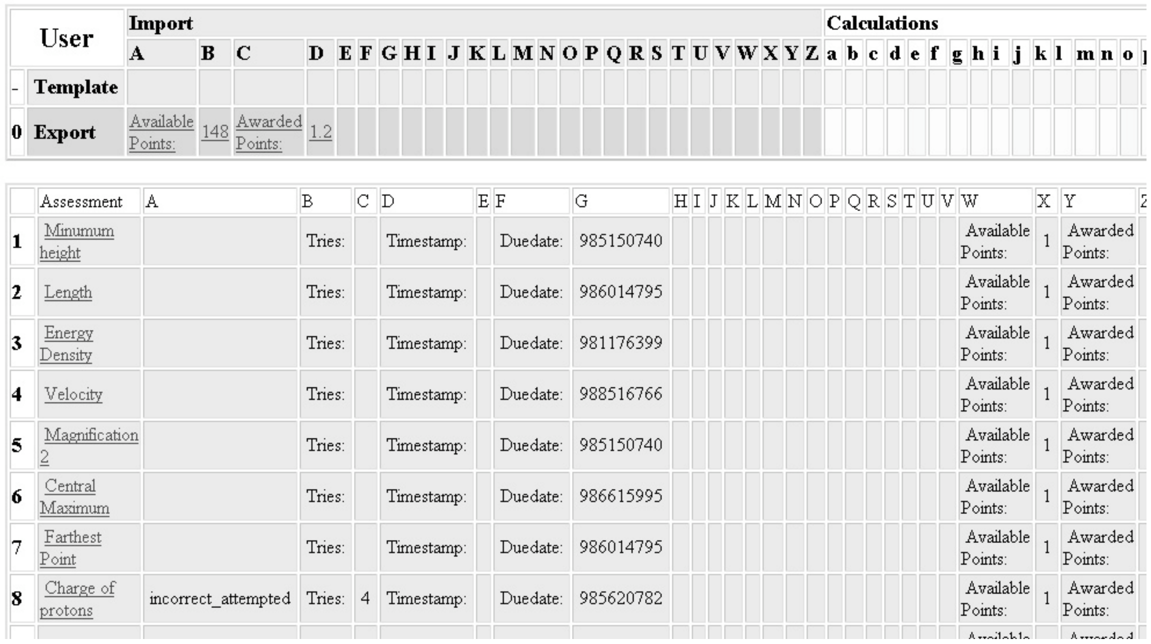
	Item	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
1	Timestamp of Last Transaction timestamp																																											
2	Number of Unsuccessful Tries [Part: 0] stores_0_tries	4																																										
3	Partial Credit Factor [Part: 0] stores_0_awarded																																											
4	Available Points [Part: 0] parameter_0_weight	1																																										
5	Problem Status [Part: 0] stores_0_solved	incorrect_attempted																																										
6	Due Date [Part: 0] parameter_0_duedate	985620782																																										
7	Maximum Number of Tries [Part: 0] parameter_0_maxtries	2																																										
8	Show Answer Date [Part: 0] parameter_0_answerdate																																											
9	[Part: 0] parameter_0_tol																																											
10	Opening Date [Part: 0] parameter_0_opendate	978726305																																										

Fig. 2.2.2 – Example Spreadsheet on Assessment Level

Student Spreadsheet

Each assessment spreadsheet exports a row into the student spreadsheet. Fig 2.2.3 shows an example student spreadsheet. The student spreadsheet exports a row to the course spreadsheet.

Fig. 2.2.3 shows the next level up spreadsheet with the exported data from this sheet.



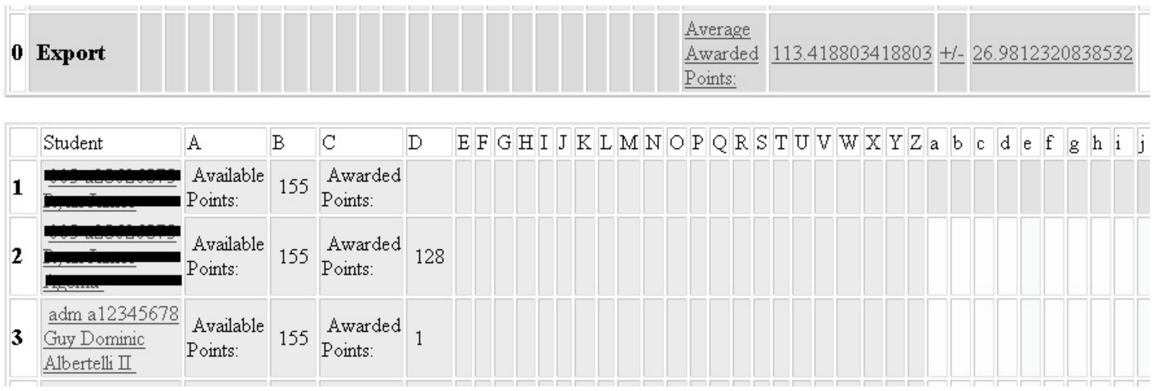
	User	Import																										Calculations														
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
-	Template																																									
0	Export	Available Points:	148	Awarded Points:	1.2																																					

	Assessment	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Minumum height		Tries:		Timestamp:		Duedate:	985150740																Available Points:	1	Awarded Points:
2	Length		Tries:		Timestamp:		Duedate:	986014795																Available Points:	1	Awarded Points:
3	Energy Density		Tries:		Timestamp:		Duedate:	981176399																Available Points:	1	Awarded Points:
4	Velocity		Tries:		Timestamp:		Duedate:	988516766																Available Points:	1	Awarded Points:
5	Magnification 2		Tries:		Timestamp:		Duedate:	985150740																Available Points:	1	Awarded Points:
6	Central Maximum		Tries:		Timestamp:		Duedate:	986615995																Available Points:	1	Awarded Points:
7	Farthest Point		Tries:		Timestamp:		Duedate:	986014795																Available Points:	1	Awarded Points:
8	Charge of protons	incorrect_attempted	Tries:	4	Timestamp:		Duedate:	985620782																Available Points:	1	Awarded Points:

Fig. 2.2.3 – Default Spreadsheet on Student Level

Course Spreadsheet

The course spreadsheet gives a summary of each students performance in the course as a whole. **Fig 2.2.4** shows the default course spreadsheet.



0	Export	Average Awarded Points:	113.418803418803	+/-	26.9812320838532

	Student	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g	h	i	j
1	[redacted]	Available Points:	155	Awarded Points:																																	
2	[redacted]	Available Points:	155	Awarded Points:	128																																
3	adm a12345678 Guy Dominic Albertelli II	Available Points:	155	Awarded Points:	1																																

Fig. 2.2.4 – Default Spreadsheet on Course Level

Spreadsheet definition

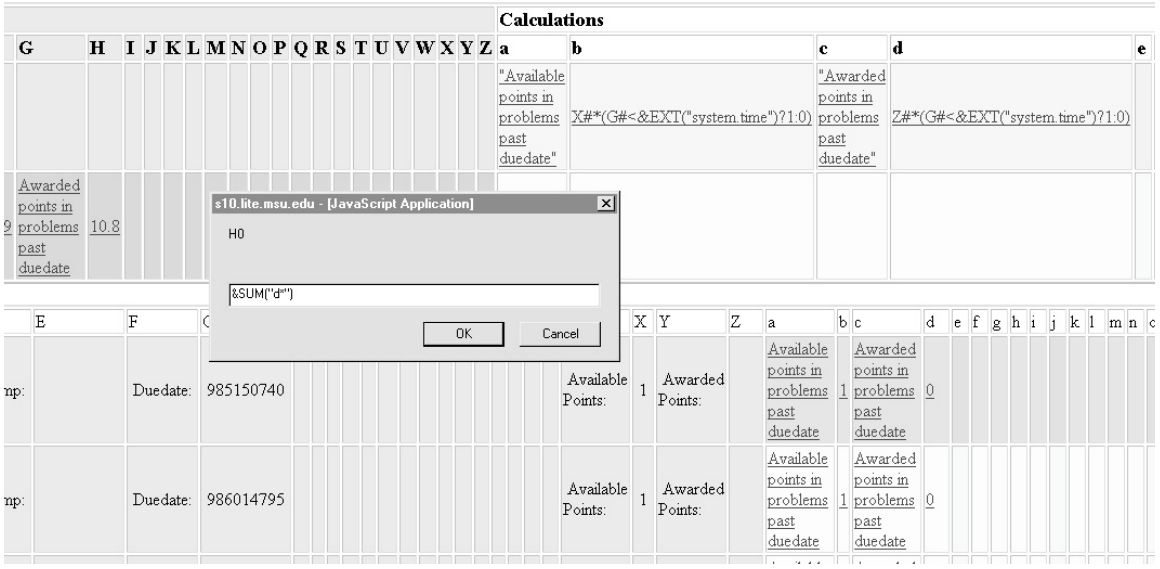
The default spreadsheets are defined via xml. **Fig. 2.2.5** shows a sample definition of a spreadsheet. However, it is possible to work with spreadsheets and never edit the xml for the spreadsheet itself. By modifying the contents of cells via the web interface and saving the results, instructors can create as complicated a spreadsheet as they like. It is now possible for users to upload spreadsheets to the library server, publish them, and set them as the default sheets for their course.

```
[www@zaphod www]$ cat /home/httpd/html/res/adm/includes/default.assesscalc
<field col=A row=0>[stores_0_solved]</field>
<field col=B row=0>'Tries:'</field>
<field col=C row=0>[stores_0_tries]</field>
<field col=D row=0>'Timestamp:'</field>
<field col=E row=0>[timestamp]</field>
<field col=F row=0>'Duedate:'</field>
<field col=G row=0>[parameter_0_duedate]</field>
<field col=W row=0>'Available Points:'</field>
<field col=X row=0>[parameter_0_weight]</field>
<field col=Y row=0>'Awarded Points:'</field>
<field col=Z row=0>[parameter_0_weight]*[stores_0_awarded]</field>
```

Fig. 2.2.5 – Default Spreadsheet on Assessment Level

Template Row

The template row of a spreadsheets will make the contents of a given column identical for each row. The contents of the template row are shown verbatim, and not evaluated. Fig 2.2.6 shows the template row in use.



Calculations

a	b	c	d
"Available points in problems past duedate"	X#*(G#<&EXT("system.time")?1:0)	"Awarded points in problems past duedate"	Z#*(G#<&EXT("system.time")?1:0)

G	H
Awarded points in problems past duedate	10.8

s10.lite.msu.edu - [JavaScript Application]

H0

&SUM("d*")

OK Cancel

F	G	W	X	Y	Z	a	b	c	d
Duedate:	985150740	Available Points:	1	Awarded Points:		Available points in problems past duedate	1	Awarded points in problems past duedate	0
Duedate:	986014795	Available Points:	1	Awarded Points:		Available points in problems past duedate	1	Awarded points in problems past duedate	0

Fig. 2.2.7 – Customized Spreadsheet on Student Level Shows a course-customized student-level spreadsheet with many of these functions, templates and wildcards in action. In the dialog window, `&SUM("d*")` is entered as the expression for cell H0, which will add up all cells in column d. The template row is used to define expressions for columns a, b, c, and d, where for b and d both the '#' wildcard and access to the EXT function is used in order to multiply the respective X and Z cells in each row with 1 or 0 depending on whether the value in G (the duedate) is smaller or larger than the system time ("?" is the standard Perl choice operator).

What goes in a cell

A cell contains either parameter data (which cannot be changed) or perl code. Additionally, cells can contain references to other cells and these references can be passed as parameters to perl functions. Parameters can be accessed via the cell they are stored in or by the parameter name.

Parameter Access

Parameters can be accessed by enclosing the parameter name in square brackets. A new feature allows the accessing of parameters by enclosing only enough information to lead to a unique parameter. "[part_0_duedate]" will give the same value as "[duedate]" if there are no other parameters which contain the string "duedate".

Addressing other cells

Cells are specified by the letter-number combination of their position in the table, for example "A5". There are also wildcards '*' and '#' which are used in ranges and templates, respectively, and symbolic names.

Valid ranges are for example "A5..C7" which is the square between cells A5 to C7, as in A5, A6, A7, B5, etc. Also, wildcards can be used, as in "A*" for column A, or "*5" for row 5. For example, &SUM("d*") will add up all cells in column d. The table below gives a brief summary of the range options.

*	all rows, all columns
B*	all rows in column B
*5	all columns in row 5
C5..F25	all cells in the rectangle between C5 and F25

The template row allows the use of "A#", which will result in "A5" in row 5, "A6" in row 6, etc.

Spreadsheet Functions

Many perl functions are available in the spreadsheet (see "The Safe Environment" below). Additionally, some spreadsheet specific functions have been defined as well. A complete list is available by executing 'perldoc loncapa/loncom/interface/lonspreadsheet.pm' on the command line. An abbreviated table appears in Fig. 2.2.7.

&NUM(*range*) – number of non-empty cells in *range*
&BIN(*low, high, range*) – number of non-empty cells in *range* with values between *low* and *high*
&SUM(*range*) – sum of the non-empty cells in *range*
&MEAN(*range*) – mean value of non-empty cells in *range*
&STDDEV(*range*) – standard deviation of non-empty cells in *range*
&PROD(*range*) – product of non-empty cells in *range*
&MAX(*range*) – maximum value of non-empty cell in *range*
&MIN(*range*) – minimum value of non-empty cells in *range*
&SUMMAX(*n ,range*) – sum of the maximum *n* non-empty cells in range
&SUMMIN(*n, range*) – sum of the minimum *n* non-empty cells in range
&EXT(*expression*) – access to EXT function in `lonnet`

Fig. 2.2.7 – Available Functions in Spreadsheet

The Safe Environment

The spreadsheet cells are evaluated in a "Safe" environment. The module Safe.pm is included in the standard systemperl RPM put out by the LON-CAPA developers, but is written by Tim Bunce and Malcolm Beattie. If you execute 'perldoc Safe' you can read the documentation for the module.

The Safe.pm module allows users to "compile and execute code in restricted compartments". We use this to allow users to implement in their spreadsheets a restricted set of perl functions, variables, and operators. No one using the spreadsheet should ever need access to the basic IO functions of Perl, for example, so these are not made available to the users.

Each cell is evaluated within the same safe space, so cells can actually contain function and declarations.

There is information the user may need which is not contained in the cells of the spreadsheet. So we poke a hole in the safe space using the Safe::Hole module. This lets us allow access to external information. Obviously this needs to be done with some degree of caution.

At the beginning of lonspreadsheet.pm we have the following code:

```
use Safe;
use Safe::Hole;
```

The function &initsheet, which is called when a spreadsheet is created or modified, defines the safe space for the execution of spreadsheet code:

```
sub initsheet {
    my $safeeval = new Safe(shift);
    my $safehole = new Safe::Hole;
    $safeeval->permit("entereval");
    $safeeval->permit(":base_math");
    $safeeval->permit("sort");
    $safeeval->deny(":base_io");
    $safehole->wrap(\&Apache::lonnet::EXT,$safeeval,'&EXT');
    my $code=<<'ENDDEFS';
.
functions and variables needed in the safe space are defined here
.
ENDDEFS
    $safeeval->reval($code);
    return $safeeval;
}
```

Change is coming

Currently the spreadsheet is scheduled to undergo major revisions by the end of the summer. The goal of these revisions is to increase the speed and add some requested functionality. We hope to add:

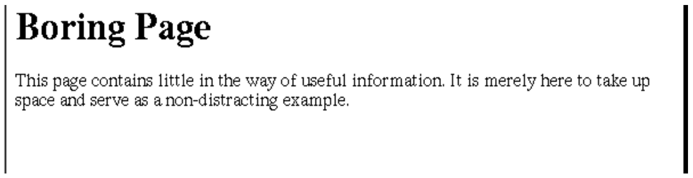
* Improved exporting of the spreadsheet date to client side spreadsheets
* Exporting of spreadsheet xml - allowing users to save spreadsheets from their classes for reuse.
* Additional spreadsheet functions.

Discussion

loncapa/loncom/interface/lonfeedback.pm handles feedback on resources. Feedback can be for course discussion of a resource or to communicate with the author of the resource. In the latter case, a screenshot of the resource and (if applicable) the students previous attempts to solve the problem.

Course discussion messages appear at the bottom of the resource being discussed. There are two types of attribution in course discussion messages. Users may choose to have their LON-CAPA id shown to everyone or they may have it shown only to instructors. Complete anonymity is not an option. If the users choose to hide their LON-CAPA id (an 'anonymous' message), they can set the name given in the [PREF] page.

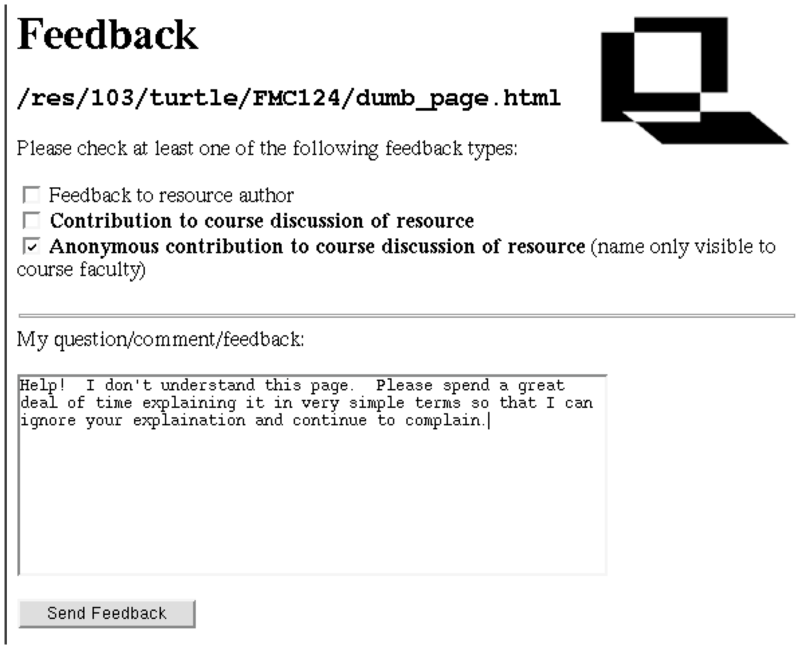
Figures 2.2.8 through **2.2.10** show a course discussion feedback cycle.



Boring Page

This page contains little in the way of useful information. It is merely here to take up space and serve as a non-distracting example.

Fig 2.2.8 - A Simple Resource



Feedback

/res/103/turtle/FMC124/dumb_page.html

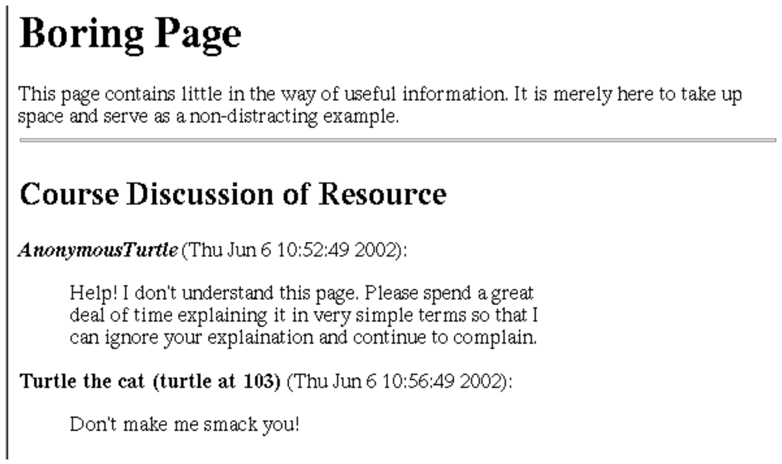
Please check at least one of the following feedback types:

- [ ] Feedback to resource author
- [ ] **Contribution to course discussion of resource**
- [✓] **Anonymous contribution to course discussion of resource** (name only visible to course faculty)

My question/comment/feedback:

```
Help!  I don't understand this page.  Please spend a great
deal of time explaining it in very simple terms so that I can
ignore your explaination and continue to complain.
```

Fig. 2.2.9 Composing Discussion Feedback



Boring Page

This page contains little in the way of useful information. It is merely here to take up space and serve as a non-distracting example.

Course Discussion of Resource

AnonymousTurtle (Thu Jun 6 10:52:49 2002):

Help! I don't understand this page. Please spend a great deal of time explaining it in very simple terms so that I can ignore your explaination and continue to complain.

Turtle the cat (turtle at 103) (Thu Jun 6 10:56:49 2002):

Don't make me smack you!

Fig 2.2.10 The Resource with Anonymous and Non-anonymous Discussion

Messaging

loncapa/loncom/interface/lonmsg.pm provides functions for sending users messages. All messaging is currently done via lonmsg.pm. The following subroutines handle the sending of various types of messages:

`author_res_msg`
Send a message to the author of a resource.

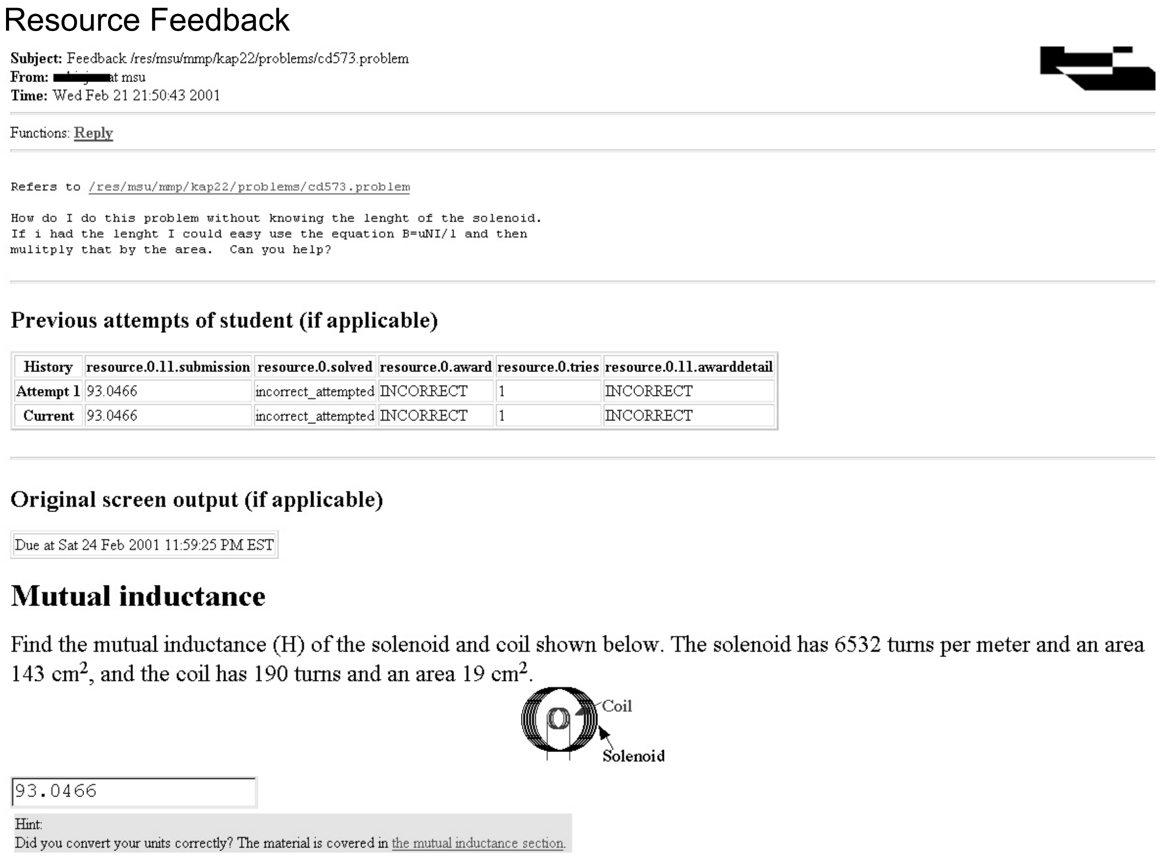
`user_crit_msg`
Send a critical message to a user. Critical messages require the user to acknowledge receipt before any other action in LON-CAPA can be taken.

`user_crit_received`
Notify the sender of a critical message that the message has been received.

`user_normal_msg`
Send a non-critical message to a user.

`statuschange`
Change the status of a message (read, replied, forwarded, etc)

The lonmsg::handler takes care of the display and sending of messages. If you have the time, communicate with yourself! Note: Critical messages are displayed on login to LON-CAPA. Some message types are only available for course coordinators.



Resource Feedback

Subject: Feedback /res/msu/mmp/kap22/problems/cd573.problem
From: [redacted] at msu
Time: Wed Feb 21 21:50:43 2001

Functions: Reply

```
Refers to /res/msu/mmp/kap22/problems/cd573.problem

How do I do this problem without knowing the lenght of the solenoid.
If i had the lenght I could easy use the equation B=uNI/l and then
mulitply that by the area.  Can you help?
```

Previous attempts of student (if applicable)

History	resource.0.11.submission	resource.0.solved	resource.0.award	resource.0.tries	resource.0.11.awarddetail
Attempt 1	93.0466	incorrect_attempted	INCORRECT	1	INCORRECT
Current	93.0466	incorrect_attempted	INCORRECT	1	INCORRECT

Original screen output (if applicable)

Due at Sat 24 Feb 2001 11:59:25 PM EST

Mutual inductance

Find the mutual inductance (H) of the solenoid and coil shown below. The solenoid has 6532 turns per meter and an area 143 cm^2, and the coil has 190 turns and an area 19 cm^2.

93.0466

Hint:
Did you convert your units correctly? The material is covered in the mutual inductance section.

Fig. 2.2.11 – Example of a Feedback

Session Three: Publication, Content Maps, Course Maps (Gerd)

Publication of a Resource

Authors can only write-access the `/~authorname/` space. They can copy resources into the resource area through the publication step, and move them back through a recover step. Authors do not have direct write-access to their resource space.

Construction space → Publication Step → **Resource space**
← Retrieve ←

During the publication step, several events will be triggered. Metadata is gathered, where a wizard manages default entries on a hierarchical per-directory base: The wizard imports the metadata (including access privileges and royalty information) from the most recent published resource in the current directory, and if that is not available, from the next directory above, etc. The Network keeps all previous versions of a resource and makes them available by an explicit version number, which is inserted between the file name and extension, for example `foo.2.html`, while the most recent version does not carry a version number (`foo.html`). Servers subscribing to a changed resource are notified that a new version is available.

Content Re-usage and Granularity

Any faculty participating in the Network can publish their own learning resources into the common pool. To that end, the Network provides a "construction space" which is only accessible to the author, and a publication process, which transfers the material to the shared pool – during the publication process, metadata about the resource is gathered, and system-wide update notification and versioning mechanisms are triggered..

Learning resources could be simple paragraphs of text, movies, applets, individualizing homework problems, etc. In addition to providing a distributed digital library with mechanisms to store and catalog these resources, the Network enables faculty to combine and sequence these resources at several levels: An instructor from Community College A could combine a text paragraph from University B with a movie from College C and an online homework problem from Publisher D, to form one page. Another instructor from High School E can take that page from Community College A and combine it with other pages into a module, unit or chapter. Those in turn can be combined into whole coursepacks. Faculty can design their own curricula from existing and newly created resources instead of having to buy into a complete off-the-shelf product.

Fig. 2.3.1 shows a general overview of the resource assembly mechanism and the different levels of content granularity supported by the current implementation of this principle. The topmost puzzle piece represents a resource at the fragment level – one GIF, one movie, one paragraph of text, one problem, or one regular web page. Attached to the resource is metadata gathered at the publication time of the resource.

Using the resource assembly tool described below, these fragments and pages can be assembled into a page. A page is a resource of the grain size which would be rendered as one page on the web and/or on the printer.

Using the same tool, fragments (which would then be rendered as standalone pages), pages and sequences can be assembled into sequences. Sequences are resources which are rendered a sequence of pages, not necessarily linear. Examples are one lesson, one chapter, or one learning cycle

On the third granularity level, fragments (rendered as standalone pages), pages, and sequences can be assembled into courses. Courses are a sequence which represents the entirety of the resources belonging to a learning unit into which learners can be enrolled. Examples are a University one-semester course, a workshop, or a High School class.

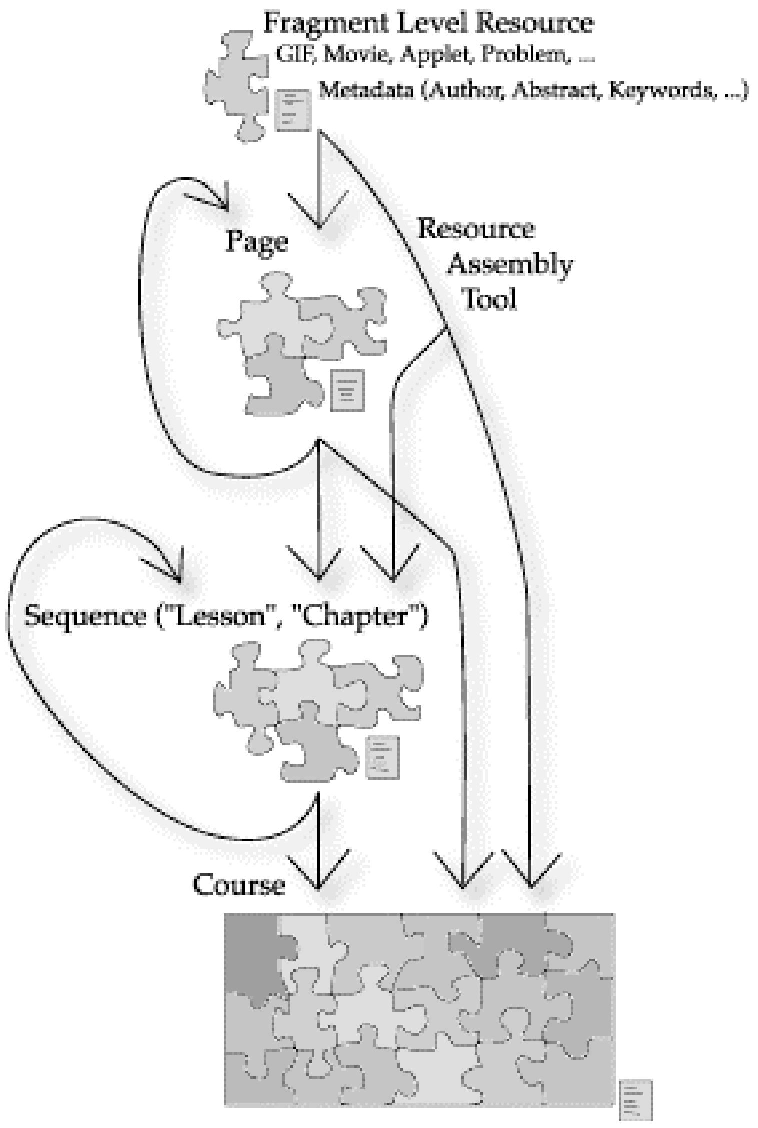

Fragment Level Resource
GIF, Movie, Applet, Problem, ...
Metadata (Author, Abstract, Keywords, ...)
Page
Resource Assembly Tool
Sequence ("Lesson", "Chapter")
Course

Fig. 2.1.1 – Resource Assembly

Maps

To increase the utility of the materials, the number of hard-coded hyperlinks between the resources should be minimized. The actual combining and sequencing is part of the system functionality and driven by external "roadmaps", which are constructed by the instructors. With this mechanism, one and the same resource can be part of different courses in different contexts. The soft-linking makes it possible to import only the desired set of resources without effectively importing additional parts another instructors resources through hard-linked menus or "next page" buttons that might resided on those resources.

Curriculum Adaptivity

Maps allow for conditional choices and branching points. The actual path through and presentation of the learning resources is determined by instructor-specified combinations of learner choices and system-generated adaptations (for example, if the learner does not pass a test, additional resources may be included). Each learner can have an individualized curriculum according to preferences, capabilities and skills.

These maps can be generated at different levels of granularity with a graphical tool, or in an automated way through custom scripts.

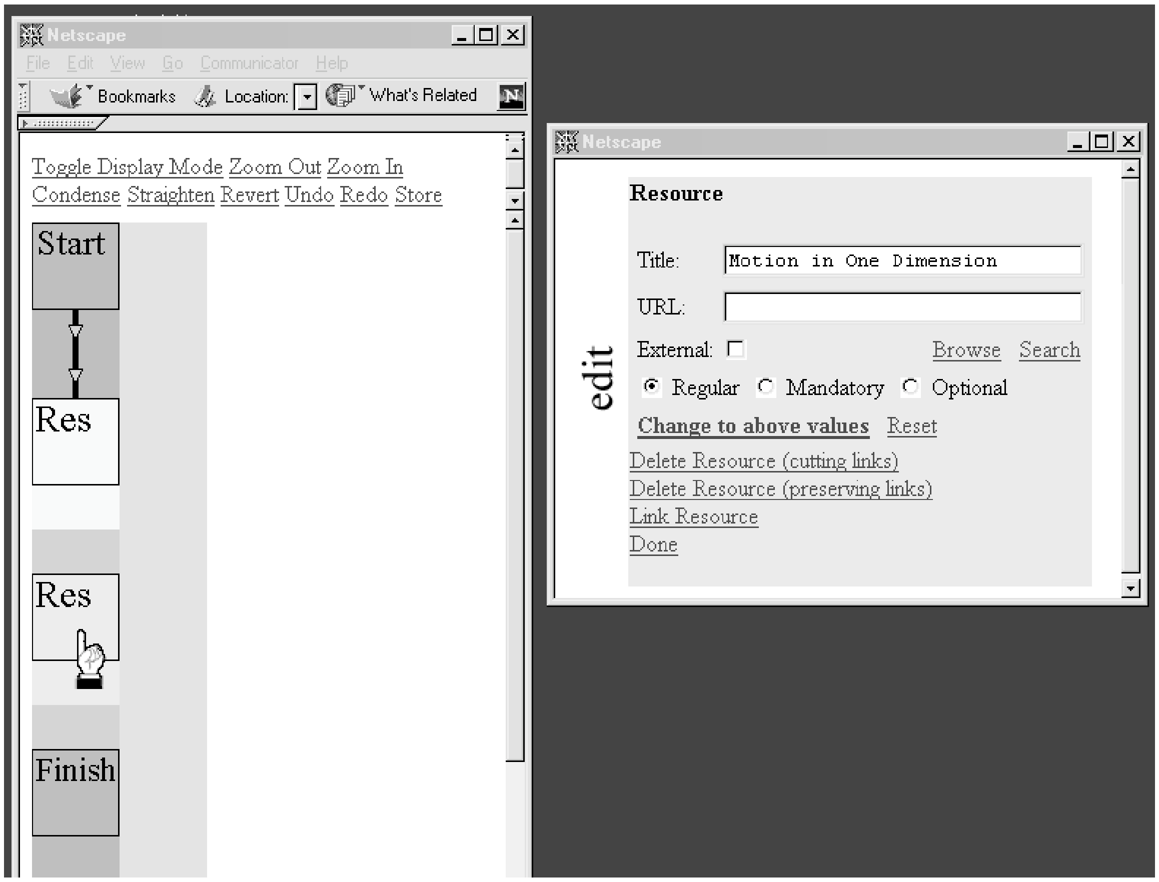
Resource Assembly Tool

The Network provides the Resource Assembly Tool as one means to generate maps. The Resource Assembly Tool provides a graphical user interface inside of a standard web browser. The current implementation is written in JavaScript.

Fig. 2.3.2 shows screenshots of the current implementation. The interface usually consists of two browser windows, one resizable one with a frameset that contains the menu and the map under construction, and a multipurpose non-resizable window that displays information and input forms.

When a new map is started, it only has a start and a finish resources. The author can then enlarge the map area and insert resources into it.

In **Fig. 2.3.2A**, the author is editing information about a resource in the map after clicking on the box representing the resource in the map. In the dialog, the author can enter a map-internal title for the resource, which is displayed to the learners when navigating the maps. In the same dialog, the author will specify the URL of the resource, which can either be internal to the Network, or any URL of a web page outside of it. For internal resources, the author can also browse the Network filesystem or search the resource metadata to locate an appropriate resource.


Netscape
File Edit View Go Communicator Help
Bookmarks Location: What's Related
Toggle Display Mode Zoom Out Zoom In
Condense Straighten Revert Undo Redo Store
Start
Res
Res
Finish
Netscape
edit
Browse Search
Change to above values Reset
Delete Resource (cutting links)
Delete Resource (preserving links)
Link Resource
Done

Resource

Title: Motion in One Dimension

URL:

External: ☐

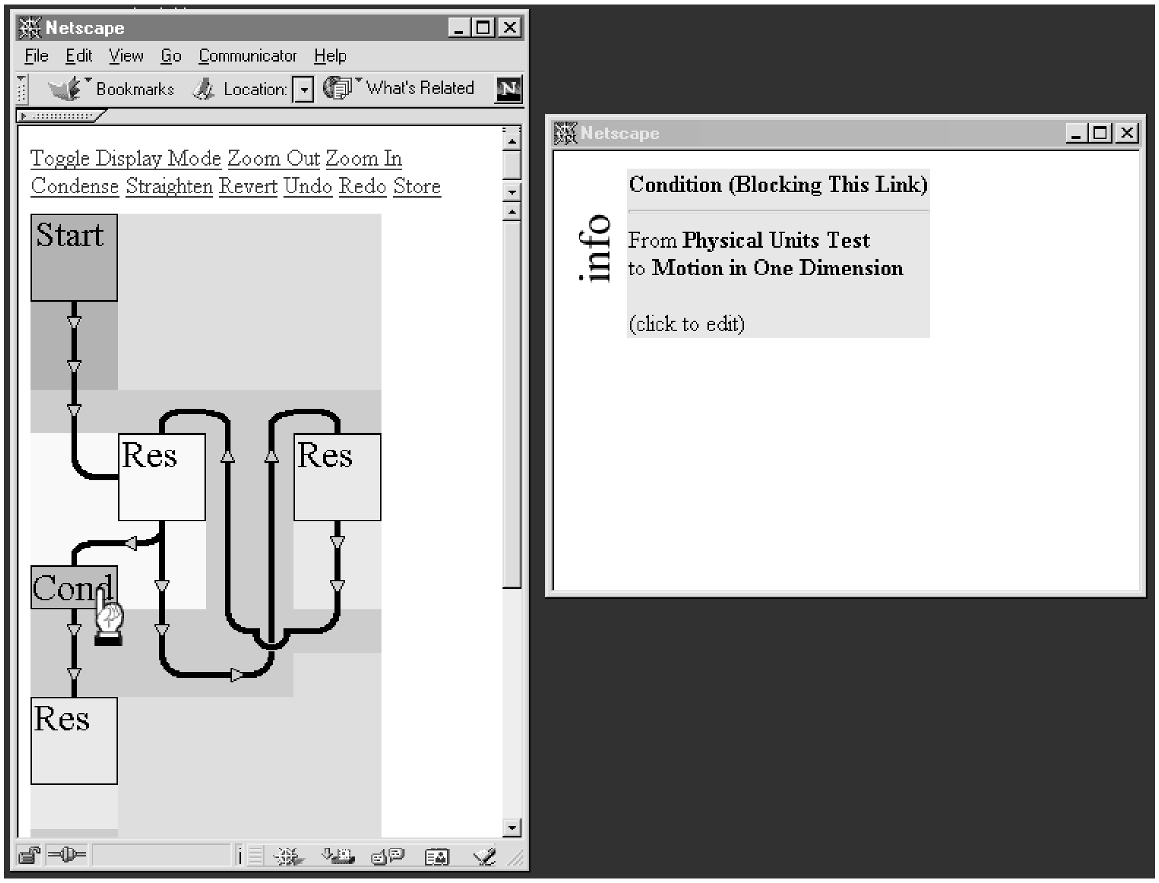
[✓ Regular] Mandatory Optional

Fig. 2.3.2A – Example, Graphical User Interface of Resource Assembly Tool

The resource priority can be chosen. A resource can be “regular,” “mandatory” or “optional.” These resource priorities are only used in book-keeping of earned points by the learners. Within the map, resources of different priorities are displayed in different colors.

The dialog also allows for two modes of removing the resource from the map: deleting it from the map including every link to and from it, and deleting it while reconnecting any links that went through the resource. As an example, resources A and B might both connect to resource C, and resource C might connect to D. When removing C from the map using the first option, A and B will not be connected to D anymore. Using the second option, in the end, A will connect to D, and B will connect to D. In the latter case, the Resource Assembly tool will also handle link conditions correctly: if A connected to C under condition 1, and C connected to D under condition 2, then in the end A will connect to D under a new condition which is (1 AND 2).

Finally, this dialog allows the author to connect the resource to another resource (or itself) through a new link. When selecting this option, the Resource Assembly Tool goes into link mode, and will link the current resource to the next resource clicked on (unless the action is cancelled).


Netscape
File Edit View Go Communicator Help
Bookmarks Location: What's Related
Toggle Display Mode Zoom Out Zoom In
Condense Straighten Revert Undo Redo Store
Start
Res
Res
Cond
Res
Netscape
info

Condition (Blocking This Link)

From **Physical Units Test**
to **Motion in One Dimension**

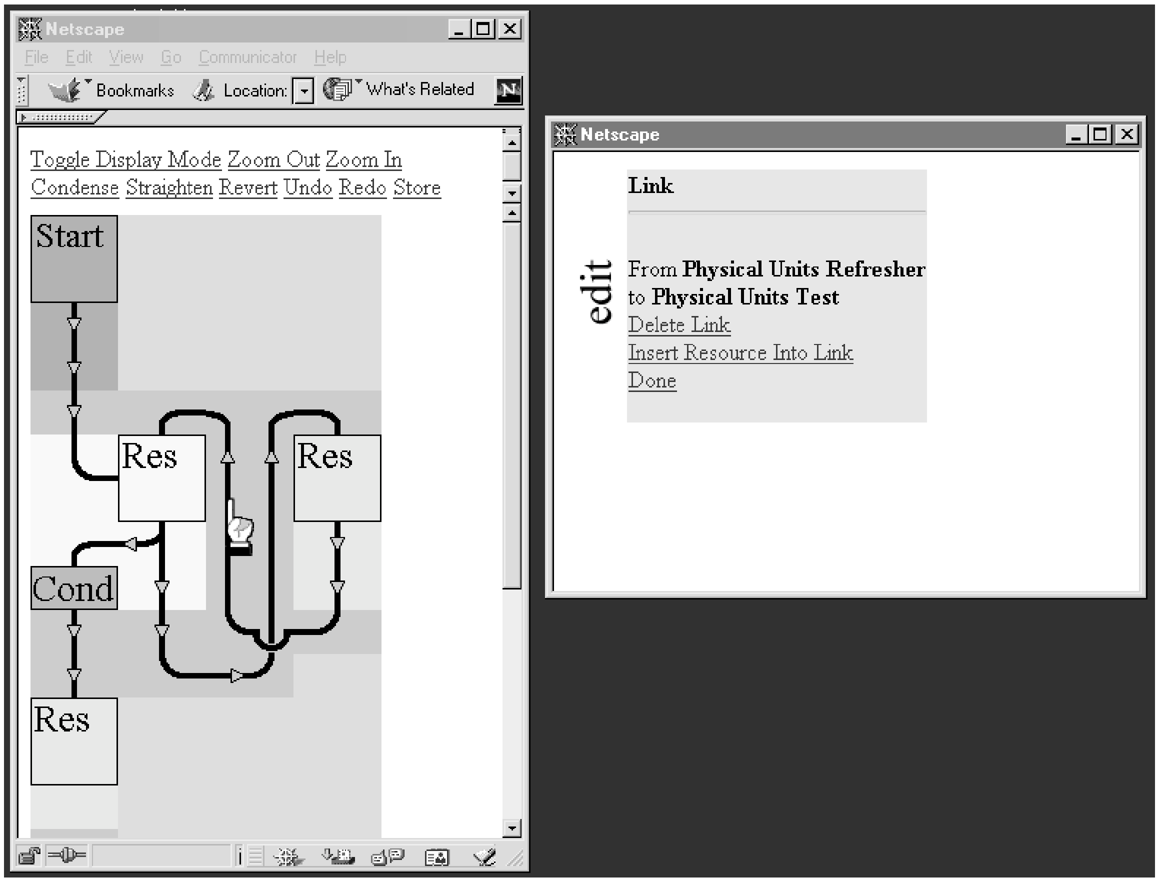
(click to edit)

Fig. 2.3.2B – Example, Graphical User Interface of Resource Assembly Tool

Fig. 2.3.2B shows the Resource Assembly Tool in info mode, that is, when no specific component of the map is edited, and if the Tool is not in link mode. In info mode, the contents of the dialog window change dynamically as the mouse is moved over the components of the map. In this case, the mouse pointer is over the link condition between two resources. The dialog window shows the titles of the connected resources, as well as the condition priority. In this scenario, the condition priority is set such that the link cannot be taken ("is blocked") if the condition is false. The condition priority can also be set such that the link is recommended if the condition is true (possibly giving the learner several options where to go next), or that the link has to be taken ("is forced") over any other possible link if the condition is true. Within the map, conditions of different priorities are displayed in different colors. If the author now were to click on the condition, the Tool would go into edit mode, and the condition could be edited.

Fig. 2.3.2C shows the Tool in edit mode for the link between the resource titles displayed. The author can remove the link, or insert a new resource into the link.

Obviously, by this mechanism, rather complex maps can be generated. These are different from binary trees, both because branches can loop back, and because branches can be re-united – in fact, most branches re-unite in the finish resources. Into each link, a condition with one of three different priorities can be attached. Whether or not a certain resource in the map can be displayed depends on whether or not it can be reached through any path along allowed links, starting with the start resource of the course. If a resource is not linked to, it is assumed to be accessible if the map which it is part of is accessible.


Netscape
File Edit View Go Communicator Help
Bookmarks
Location:
What's Related
Toggle Display Mode Zoom Out Zoom In
Condense Straighten Revert Undo Redo Store
Start
Res
Res
Cond
Res
Netscape
edit

Link

From **Physical Units Refresher**
to **Physical Units Test**
Delete Link
Insert Resource Into Link
Done

Fig. 2.3.2C – Example, Graphical User Interface of Resource Assembly Tool

Map Representation and Storage Format

Fig. 2.3.3 shows the XML representation of the resource map constructed in **Fig. 2.3.2**, which is the format in which maps are stored. In the figure, however, additional graphical map layout information generated by the Resource Assembly Tool is not displayed. This graphical information is optional to re-generate the same graphical layout when the map is brought up again in the Resource Assembly Tool, and is not needed for any other system functionality.

Maps can be generated by tools other than the Resource Assembly Tool. In particular, an author might have some other representation of a course sequence, which can be converted into a map using scripts. If this map then were to be brought up in the Resource Assembly Tool, the Tool would automatically generate a graphical layout for it. Each entry of the map (resources, conditions and links) is stored in a separate tag.

Resources and conditions have to have unique ID numbers. These numbers are automatically generated by the Resource Assembly Tool when the entry is first created, or added to the entries when a map generated outside the Resource Assembly Tool is first retrieved. They can also be assigned by custom scripts or added in by hand.

In this example, **Fig. 2.3.3**, entry 1 is the start resource of the map – when this map is accessed, the source (src) URL of this tag will be the first resource rendered. Entry 2 is the finish resource of this map. This resource will be the last resource in the sequence of resources. Entry 6 is a problem resource with the given URL and title, as well as the priority “mandatory”. Entry 19 is a condition, which is used by the link between entries 6, the problem, and 9, a sequence. *The final syntax for conditions has not yet been determined.*

```
<map>
<resource id="1"
          src="/res/msu/korte/phy231welcome.html"
          type="start"
          title="Start"></resource>
<resource id="2"
          src="" type="finish"
          title="Finish"></resource>
<resource id="6"
          src="/res/msu/korte/tests/units.problem"
          type="mandatory"
          title="Physical Units Test"></resource>
<resource id="9"
          src="/res/msu/korte/chapters/onedim.sequence"
          title="Motion in One Dimension"></resource>
<resource id="11"
          src="/res/msu/bauer/bridges/units.sequence"
          title="Physical Units Refresher"></resource>
<condition id="19"
         type="stop"
         value="user.assessments[this./res/msu/korte/tests/units.problem].status=solved">
  </condition>
<link from="1" to="6"></link>
<link from="6" to="9" condition="19"></link>
<link from="6" to="11"></link>
<link from="11" to="6"></link>
</map>
```

Fig. 2.3.3 – XML representation of the map in Fig. 2.3.2C (non-graphical information only).

Example of Nested Maps

Fig. 2.3.4 shows the XML representation of three maps which are imported into each other. **Fig. 2.3.4B** is the sequence that is referenced as resource 9 in the course map **Fig. 2.3.4A**. In the resulting map, the entry point of resource 9 in **Fig. 2.3.4A** is in fact the entry point of the start resource of **Fig. 2.3.4B**, namely, resource 1 there. The exit point of resource 9 in **Fig. 2.3.4A** is the exit point of the finish resource of **Fig. 2.3.4B**, namely, resource 2 there.

Fig. 2.3.4C is the page which is referenced as resource 24 in **Fig. 2.3.4B**.

A course can easily contain several hundreds of these nested maps. Since the accessibility of each individual resource in the course depends on the state of all possible paths linking it to the start resource of the course across all intermediate maps, the computation and disk-I/O effort per single transaction could quickly become prohibitive. Thus, all maps and conditions are compiled into a pre-processed binary data structure at the start of a session.

```
<map>
<resource id="1" src="" type="start" title="Start"></resource>
<resource id="2" src="" type="finish" title="Finish"></resource>
<resource id="5" src="/res/msu/korte/tests/pretest.problem" type="mandatory"
  title="Pretest"></resource>
<resource id="9" src="/res/msu/korte/parts/part1.sequence" type="mandatory"
  title="Part 1"></resource>
<resource id="11" src="/res/msu/korte/tests/midterm.sequence" type="mandatory"
  title="Midterm"></resource>
<resource id="15" src="/res/msu/korte/parts/part2.sequence" type="mandatory"
  title="Part 2"></resource>
<condition id="19" type="stop"
  value="user.assessments[this./msu/korte/tests/pretest.problem].status=solved">
  </condition>
<resource id="20" src="/res/msu/korte/refresh/refresher.sequence"
  title="Refresher"></resource>
<resource id="29" src="/res/msu/korte/tests/final.sequence" type="mandatory"
  title="Final Exam"></resource>
<condition id="30" type="stop"
  value="user.assessments[this./msu/korte/tests/midterm.sequence].percent>60">
  </condition>
<resource id="36" src="/res/msu/korte/refresh/review.sequence"
  title="Review"></resource>
<condition id="43" type="force"
  value="user.assessments[this./msu/korte/tests/midterm.sequence].percent<10">
  </condition>
<resource id="58" src="/res/msu/korte/chapters/applications.sequence" type="optional"
  title="Applications"></resource>
<condition id="70" type="stop"
  value="user.assessments[this./msu/korte/tests/final.sequence].percent>60">
  </condition>
<link from="1" to="5"></link>
<link from="9" to="11"></link>
<link from="11" to="15" condition="30"></link>
<link from="5" to="9" condition="19"></link>
<link from="5" to="20"></link>
<link from="20" to="5"></link>
<link from="11" to="36" condition="43"></link>
<link from="36" to="9"></link>
<link from="36" to="11"></link>
<link from="15" to="29"></link>
<link from="29" to="2" condition="70"></link>
<link from="11" to="11"></link>
</map>
```

Fig. 2.3.4A – Example of a course map that has nested sequences

```
<map>
<resource id="1" src="" type="start" title="Start"></resource>
<resource id="2" src="" type="finish" title="Finish"></resource>
<resource id="5" src="/res/msu/korte/parts/part1intro.html"
  title="Part 1 Introduction"></resource>
<resource id="6" src="/res/msu/korte/parts/part1dir.xml" title="Directions"></resource>
<resource id="12" src="/res/msu/korte/tests/part11.problem" title="Problem 1"></resource>
<resource id="13" src="/res/msu/korte/tests/part13.problem" title="Problem 3"></resource>
<resource id="19" src="/res/msu/korte/tests/part12.problem" title="Problem 2"></resource>
<resource id="24" src="/res/msu/korte/parts/summary.page" title="Summary"></resource>
<condition id="47" type="stop"
  value="user.assessments[this./msu/korte/tests/part11.problem].status=solved">
  </condition>
<condition id="48" type="stop"
  value="user.assessments[this./msu/korte/tests/part12.problem].status=solved">
  </condition>
<condition id="49" type="stop"
  value="user.assessments[this./msu/korte/tests/part13.problem].status=solved">
  </condition>
<link from="5" to="6"></link>
<link from="1" to="5"></link>
<link from="6" to="12"></link>
<link from="6" to="13"></link>
<link from="6" to="19"></link>
<link from="12" to="24" condition="47"></link>
<link from="19" to="24" condition="48"></link>
<link from="13" to="24" condition="49"></link>
<link from="24" to="2"></link>
</map>
```

Fig. 2.3.4B – Example of a sequence (`part1.sequence`) that has nested pages

```
<map>
<resource id="1" src="" type="start" title="Start"></resource>
<resource id="2" src="" type="finish" title="Finish"></resource>
<resource id="5" src="/res/msu/smith/racecar.problem"></resource>
<resource id="6" src="/res/msu/smith/toofast.html"></resource>
<resource id="8" src="/res/msu/smith/tooslow.html"></resource>
<resource id="15" src="/res/msu/smith/accelerate.html"></resource>
<condition id="40" type="force"
  value="user.assessments[this./msu/smith/racecar.problem].status=solved"></condition>
<condition id="41" type="stop"
  value="user.assessments[this./msu/smith/racecar.problem].answer=friction"></condition>
<condition id="42" type="stop"
  value="user.assessments[this./msu/smith/racecar.problem].answer=sliding"></condition>
<condition id="43" type="stop"
value="user.assessments[this./msu/smith/racecar.problem].answer=nonconstant"></condition>
<link from="1" to="5"></link>
<link from="5" to="6" condition="41"></link>
<link from="5" to="8" condition="42"></link>
<link from="5" to="15" condition="43"></link>
<link from="6" to="2"></link>
<link from="8" to="2"></link>
<link from="15" to="2"></link>
<link from="5" to="2" condition="40"></link>
</map>
```

Fig. 2.3.4C – Example of a page (`summary.page`)

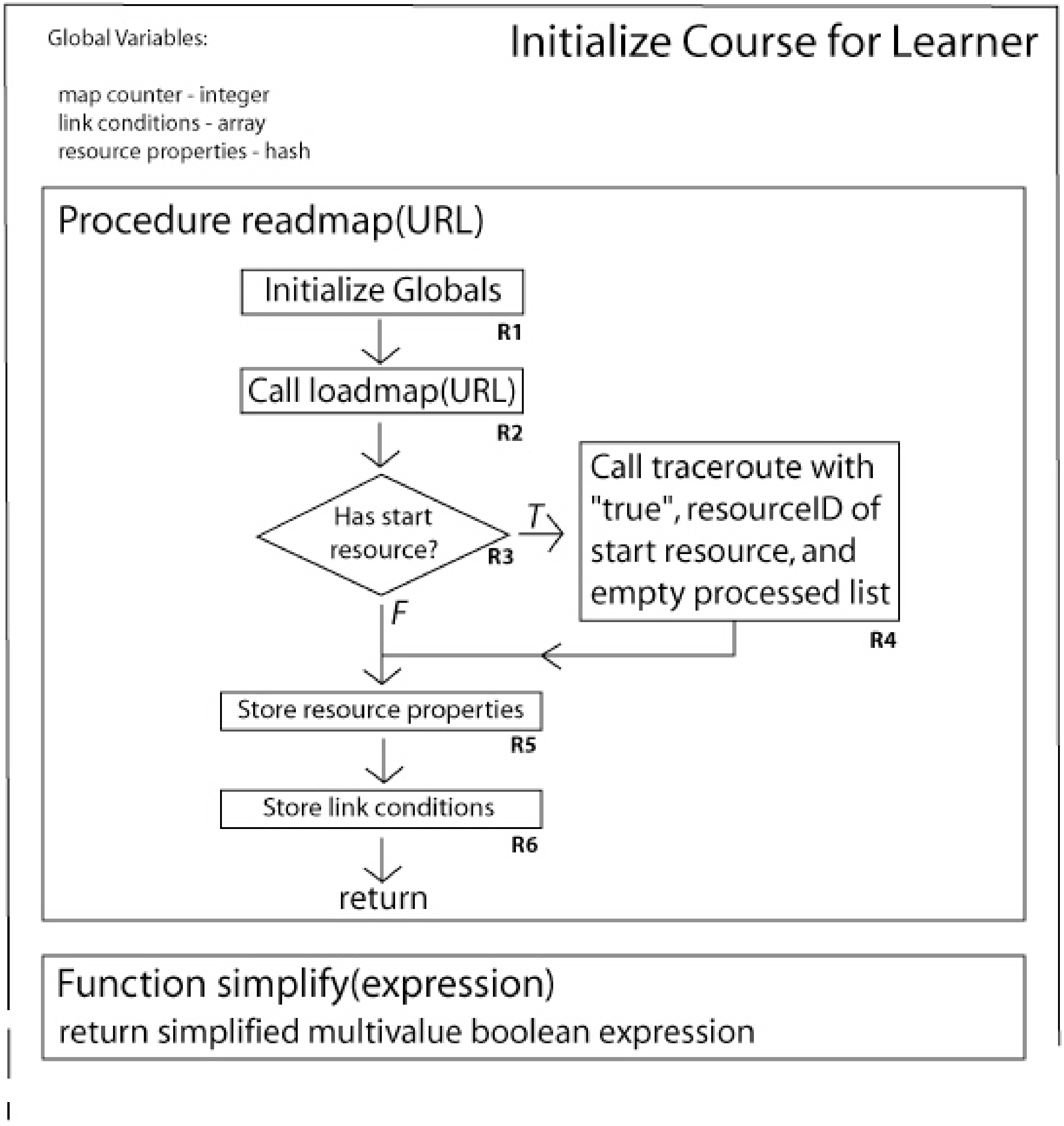

Global Variables:
Initialize Course for Learner
map counter - integer
link conditions - array
resource properties - hash
Procedure readmap(URL)
Initialize Globals
R1
Call loadmap(URL)
R2
Has start resource?
R3
T
F
Call traceroute with "true", resourceID of start resource, and empty processed list
R4
Store resource properties
R5
Store link conditions
R6
return
Function simplify(expression)
return simplified multivalue boolean expression

Fig. 2.3.5 – Flow chart of the course initialization routine run when a learner first accesses a course during a session (see Figs. 2.1.6A and 2.1.8A for the procedures `loadmap` and `traceroute`)

Initialization of a Course for a Learner

When a learner first enters a course during a session, the system will initialize this course for the learner. In particular, at this point, the course map and all nested (embedded) maps and resources are evaluated, and the information is compiled into two binary structures, which are stored with the session information: the resource properties hash, and the link conditions array. This information will be used over the duration of the session for several purposes: navigation (which resource is the next, which one the previous?), for access control (can the resource be reached under the link conditions given the current state of the student?), and to register assessment results within the context of a certain course and map (there might be several instances of the same problem resource within a course).

Evaluation of the Map Structure for a Course

The URL of the course is passed to the procedure `readmap` (**Fig. 2.3.5**). Procedure `readmap` first initializes the resource properties as an empty hash, seeds the link conditions array with a 0th element, which is set to "true", priority "normal", and sets the map counter to 0 (**Fig. 2.3.5, Step R1**). While the resource properties hash, the link conditions array and the map counter are global variable of the initialization process, all other variables are local to the procedures (an important property for these routines to run recursively). The procedure `readmap` then calls procedure `loadmap` for the URL of the course (**Fig. 2.3.5, Step R2**).

Figs. 2.1.6, 2.1.7 show a dump of excerpts of the binary structure generated in `loadmap` for the nested maps of example **Fig. 2.3.4**.

Procedure `loadmap` (**Fig. 2.3.6A**) first checks if the map URL has already been processed (multiple inclusion of the same map in a course structure) (**Fig. 2.3.6A, Step L1**) – if it was, it has been assigned a map counter value in the resource properties hash. If the map has been processed, there is no need to process it again, and `loadmap` returns.

If the map has not been processed yet, the map counter is incremented and the map is registered under the current value in the resource properties hash (**Fig. 2.3.6A, Step L2**). The file is then opened (**Fig. 2.3.6A, Step L3**), which might entail prior replication, and the contents are parsed. If there are no further entries, `loadmap` returns (**Fig. 2.3.6A, Step L4**).

The new entry tag is then read (**Fig. 2.3.6A, Step L5**) and the type is determined (**Fig. 2.3.6A, Step L6**).

If the entry is a resource (**Step L7**), a resource ID is formed by combining the map counter and the resource ID within the map. For example, the "Part I Introduction" resource of part1.sequence (**Fig. 2.3.4B**) was assigned the resource ID 2.5, since it has the internal resource ID 5 in the 2nd map processed (see **Fig. 2.3.6B** under "ids_"). If the same URL is found again, additional IDs are assigned to it. It is necessary to store the IDs under the URL in the resource properties hash for reverse lookup if a user simply requests a URL. If the resource is a start or finish resources, the resource ID is registered as the start or finish resource of the map, respectively (**Fig. 2.3.6B**, "map_start_", "map_finish_"). The properties of the resource (URL, Title, Priority, etc) are now stored under the resource ID, see for example **Fig. 2.3.6B** "title_2.5".

If the resource is not a map itself (**Fig. 2.3.6A**, **Step L8**), the next entry is read. Otherwise, procedure `loadmap` calls itself recursively to process that map (**Step L9**).

If in **Step L6**, the type of the entry was determined to be a condition, a condition ID is formed (**Step L10**) by again combining the map counter with the internal ID. The condition is also added to the end of the condition array (see **Fig. 10**), which is a compilation of all conditions in the course (**Step L11**). The conditions in this array are evaluated when a transaction occurs that could change the state of the student, and the state of each condition is stored by the index number in the session environment. A reference to the index number in the condition array is stored under the condition ID (**Fig. 2.3.6D**, "condid_").

If the entry is a link (**Step L6**), a link ID is generated (**Step L12**). This ID is formed by combining the map counter and another counter which is incremented for every new link within the map. Under this ID, the IDs of the originating and the destination resource of the link are stored, as well as that of the link condition (**Fig. 2.3.6D**). For the originating resource, in **Step L13** the link ID is added to the list of outgoing links (**Fig. 2.3.6C**, "to_"), and for the destination resource, the link ID is added to the list of incoming links (**Fig. 2.3.6C**, "from_").

After the last entry has been processed, procedure `loadmap` returns. After the last map has been processed, the original course-level instance of `loadmap` returns to `readmap` (**Fig. 2.3.5**, **Step R2**).

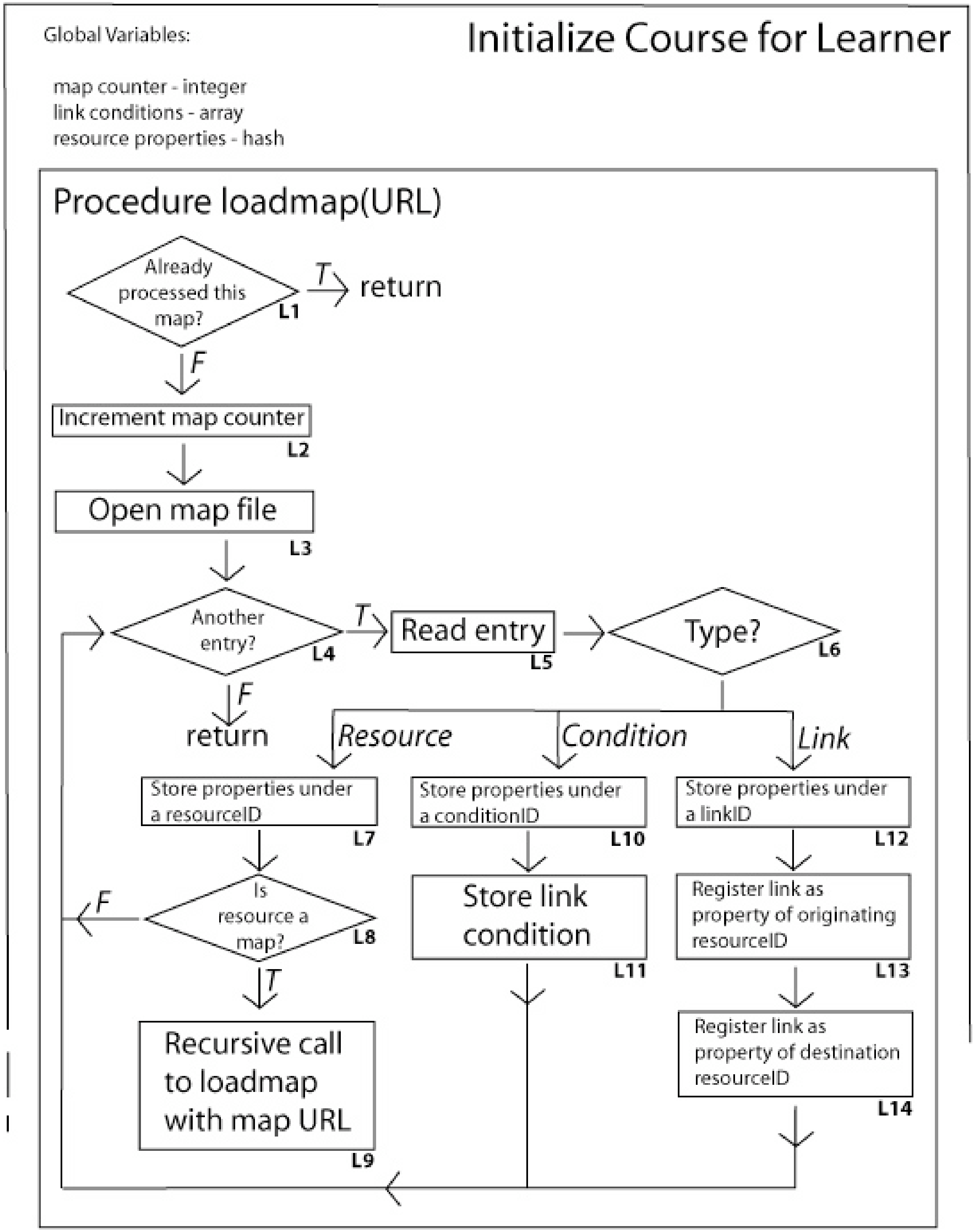

Global Variables:
Initialize Course for Learner
map counter - integer
link conditions - array
resource properties - hash
Procedure loadmap(URL)
Already processed this map?
L1
T
return
F
Increment map counter
L2
Open map file
L3
Another entry?
L4
T
Read entry
L5
Type?
L6
F
return
Resource
Condition
Link
Store properties under a resourceID
L7
Store properties under a conditionID
L10
Store properties under a linkID
L12
Is resource a map?
L8
F
T
Recursive call to loadmap with map URL
L9
Store link condition
L11
Register link as property of originating resourceID
L13
Register link as property of destination resourceID
L14

Fig. 2.3.6A – Flow chart of procedure `loadmap`

```
ids_/res/msu/korte/chapters/applications.sequence: 1.58
ids_/res/msu/korte/parts/part1.sequence: 1.9
ids_/res/msu/korte/parts/part1dir.xml: 2.6
ids_/res/msu/korte/parts/part1intro.html: 2.5
ids_/res/msu/korte/parts/part2.sequence: 1.15
ids_/res/msu/korte/parts/summary.page: 2.24
ids_/res/msu/korte/refresh/refresher.sequence: 1.20
ids_/res/msu/korte/refresh/review.sequence: 1.36
ids_/res/msu/korte/tests/final.sequence: 1.29
ids_/res/msu/korte/tests/midterm.sequence: 1.11
ids_/res/msu/korte/tests/part11.problem: 2.12
ids_/res/msu/korte/tests/part12.problem: 2.19
ids_/res/msu/korte/tests/part13.problem: 2.13
ids_/res/msu/korte/tests/pretest.problem: 1.5
ids_/res/msu/smith/accelerate.html: 3.15
ids_/res/msu/smith/racecar.problem: 3.5
ids_/res/msu/smith/toofast.html: 3.6
ids_/res/msu/smith/tooslow.html: 3.8
…
map_start_/res/msu/korte/foo.sequence: 1.1
map_start_/res/msu/korte/parts/part1.sequence: 2.1
map_start_/res/msu/korte/parts/summary.page: 3.1
…
map_finish_/res/msu/korte/foo.sequence: 1.2
map_finish_/res/msu/korte/parts/part1.sequence: 2.2
map_finish_/res/msu/korte/parts/summary.page: 3.2
…
title_1.11: Midterm
title_1.15: Part 2
title_1.20: Refresher
title_1.29: Final Exam
title_1.36: Review
title_1.5: Pretest
title_1.58: Applications
title_1.9: Part 1
title_2.12: Problem 1
title_2.13: Problem 3
title_2.19: Problem 2
title_2.24: Summary
title_2.5: Part 1 Introduction
title_2.6: Directions
…
```

Fig. 2.3.6B – Dump of the resource properties hash. Excerpt of the resource properties gathered in procedure `loadmap`

```
to_1.1: 1.1
to_1.11: 1.3,1.7,1.12
to_1.15: 1.10
to_1.20: 1.6
to_1.29: 1.11
to_1.36: 1.8,1.9
to_1.5: 1.4,1.5
to_1.9: 1.2
to_2.1: 2.2
to_2.12: 2.6
to_2.13: 2.8
to_2.19: 2.7
to_2.24: 2.9
to_2.5: 2.1
to_2.6: 2.3,2.4,2.5
to_3.1: 3.1
to_3.15: 3.7
to_3.5: 3.2,3.3,3.4,3.8
to_3.6: 3.5
to_3.8: 3.6
…
from_1.11: 1.2,1.9,1.12
from_1.15: 1.3
from_1.2: 1.11
from_1.20: 1.5
```

```
from_1.29: 1.10
from_1.36: 1.7
from_1.5: 1.1,1.6
from_1.9: 1.4,1.8
from_2.12: 2.3
from_2.13: 2.4
from_2.19: 2.5
from_2.2: 2.9
from_2.24: 2.6,2.7,2.8
from_2.5: 2.2
from_2.6: 2.1
from_3.15: 3.4
from_3.2: 3.5,3.6,3.7,3.8
from_3.5: 3.1
from_3.6: 3.2
from_3.8: 3.3
…
```

Fig. 2.3.6C – Dump of the resource properties hash. Excerpt of information gathered about links between resources in subroutine `loadmap`.

```
goesto_1.1: 1.5           comesfrom_1.1: 1.1        undercond_1.1: 0
goesto_1.10: 1.29         comesfrom_1.10: 1.15      undercond_1.10: 0
goesto_1.11: 1.2          comesfrom_1.11: 1.29      undercond_1.11: 1.70
goesto_1.12: 1.11         comesfrom_1.12: 1.11      undercond_1.12: 0
goesto_1.2: 1.11          comesfrom_1.2: 1.9        undercond_1.2: 0
goesto_1.3: 1.15          comesfrom_1.3: 1.11       undercond_1.3: 1.30
goesto_1.4: 1.9           comesfrom_1.4: 1.5        undercond_1.4: 1.19
goesto_1.5: 1.20          comesfrom_1.5: 1.5        undercond_1.5: 0
goesto_1.6: 1.5           comesfrom_1.6: 1.20       undercond_1.6: 0
goesto_1.7: 1.36          comesfrom_1.7: 1.11       undercond_1.7: 1.43
goesto_1.8: 1.9           comesfrom_1.8: 1.36       undercond_1.8: 0
goesto_1.9: 1.11          comesfrom_1.9: 1.36       undercond_1.9: 0
goesto_2.1: 2.6           comesfrom_2.1: 2.5        undercond_2.1: 0
goesto_2.2: 2.5           comesfrom_2.2: 2.1        undercond_2.2: 0
goesto_2.3: 2.12          comesfrom_2.3: 2.6        undercond_2.3: 0
goesto_2.4: 2.13          comesfrom_2.4: 2.6        undercond_2.4: 0
goesto_2.5: 2.19          comesfrom_2.5: 2.6        undercond_2.5: 0
goesto_2.6: 2.24          comesfrom_2.6: 2.12       undercond_2.6: 2.47
goesto_2.7: 2.24          comesfrom_2.7: 2.19       undercond_2.7: 2.48
goesto_2.8: 2.24          comesfrom_2.8: 2.13       undercond_2.8: 2.49
goesto_2.9: 2.2           comesfrom_2.9: 2.24       undercond_2.9: 0
goesto_3.1: 3.5           comesfrom_3.1: 3.1        undercond_3.1: 0
goesto_3.2: 3.6           comesfrom_3.2: 3.5        undercond_3.2: 3.41
goesto_3.3: 3.8           comesfrom_3.3: 3.5        undercond_3.3: 3.42
goesto_3.4: 3.15          comesfrom_3.4: 3.5        undercond_3.4: 3.43
goesto_3.5: 3.2           comesfrom_3.5: 3.6        undercond_3.5: 0
goesto_3.6: 3.2           comesfrom_3.6: 3.8        undercond_3.6: 0
goesto_3.7: 3.2           comesfrom_3.7: 3.15       undercond_3.7: 0
goesto_3.8: 3.2           comesfrom_3.8: 3.5        undercond_3.8: 3.40
…                         …                         …
                                                    condid_1.19: 8
                                                    condid_1.30: 9
                                                    condid_1.43: 10
                                                    condid_1.70: 11
                                                    condid_2.47: 5
                                                    condid_2.48: 6
                                                    condid_2.49: 7
                                                    condid_3.40: 1
                                                    condid_3.41: 2
                                                    condid_3.42: 3
                                                    condid_3.43: 4
                                                    …
```

Fig. 2.3.6D – Dump of the resource properties hash. Excerpt of information gathered about links and link conditions between resources in subroutine `loadmap`.

```
0 : true:normal
1 : user.assessments[this./msu/smith/racecar.problem].status=solved:force
2 : user.assessments[this./msu/smith/racecar.problem].answer=friction:stop
3 : user.assessments[this./msu/smith/racecar.problem].answer=sliding:stop
4 : user.assessments[this./msu/smith/racecar.problem].answer=nonconstant:stop
5 : user.assessments[this./msu/korte/tests/part11.problem].status=solved:stop
6 : user.assessments[this./msu/korte/tests/part12.problem].status=solved:stop
7 : user.assessments[this./msu/korte/tests/part13.problem].status=solved:stop
8 : user.assessments[this./msu/korte/tests/pretest.problem].status=solved:stop
9 : user.assessments[this./msu/korte/tests/midterm.sequence].percent>60:stop
10 : user.assessments[this./msu/korte/tests/midterm.sequence].percent<10:force
11 : user.assessments[this./msu/korte/tests/final.sequence].percent>60:stop

•••
```

Fig. 2.3.7 – Excerpt of the dump of the condition array constructed in procedure `loadmap` *(the final syntax of conditions has not yet been determined)*

Paths and Path Conditions

The next mayor step will be to determine all possible paths and conditions leading up to a resource for access control.

`readmap` checks if the course has a start resource from its map_start entry in the resource properties (Step R2), and if does not, continue to store the two global binary data structures (**Steps R5,R6**) – in this special case, all resources which are part of any maps in the course are assumed to be accessible.

If the course has a start resource, `readmap` calls the procedure `traceroute` (**Fig. 2.3.8A**) with the following parameters (**Step R4**): The cumulative condition along this path or route so far is set to "true" (the map is accessible); the resource ID of the start resource of the course map; and an empty list for all resources processed so far along this route. It is again important to note that all variables except the global binary structures are local to `traceroute`, since `traceroute` will recursively call itself whenever there is a branching to follow all possible paths of the maps.

`traceroute` will establish a section within the resource properties hash that builds up all conditions leading up to a resource. **Fig. 2.3.8B** shows an excerpt of the final result. For example, resource 2.5, the introduction to part 1, can be reached under condition 8 (see **Fig. 2.3.7**), meaning, after solving the pretest problem.

`traceroute` first checks if the resource has already been processed on this route by its resource ID (**Fig. 2.3.8A**, **Step T1**) – this test avoids that `traceroute` runs into endless loops when the links on the map loop. Next, the resource ID is added to the list of processed resources on this route (**Step T2**).

The resource conditions are now OR'd with the cumulative conditions on this route (**Step T3**) – the route represents another way of getting to the resource. A small routine with simplification rules for boolean expressions is called to simplify the potentially very long expression.

In the next step, it is determined if the resource is itself a map (**Step T4**). If it is, the exit route conditions can differ from the entry route condition by all additional conditions along the paths in the embedded maps (for non-map resources, entry and exit route conditions are the same). If however the embedded map does not have a start resource (**Step T5**), that is not the case – again, the missing of entry point to an embedded link structure is interpreted as the resources being openly accessible.

If the embedded map does have a start resource, `traceroute` is called recursively with the current route conditions, the ID of the start resource of that map, and the list of already processed resource IDs (**Step T6**). Upon return, if the embedded map does not have a finish resource, the entry and exit conditions of this map are assumed to be the same (**Step T7**). If the map had a finish resource, the route condition so far is set to the resource condition of the finish resource of the embedded map (**Step T8**) – in order go on from here, the user would have had to reach the finish resource of the embedded map.

Now the route conditions are correctly set for exiting the resource and going on from here. `traceroute` now loops over all outgoing links of the resource (**Step T9**). If the link does have a link condition (**Step T10**), then for this path, the cumulative route condition so far AND the link condition (**Step T11**). If there is no link condition, then there is no change in route conditions (**Step T12**).

To further process the routes along this link, `traceroute` is called recursively with the resource ID of the destination resource of the link, the new route conditions, and the list of already processed resources (**Step T13**). `traceroute` returns after processing the last outgoing link of the resource it had been called for.

Fig. 2.3.8B shows part of the output of `traceroute` for the example **Fig. 2.3.4**.

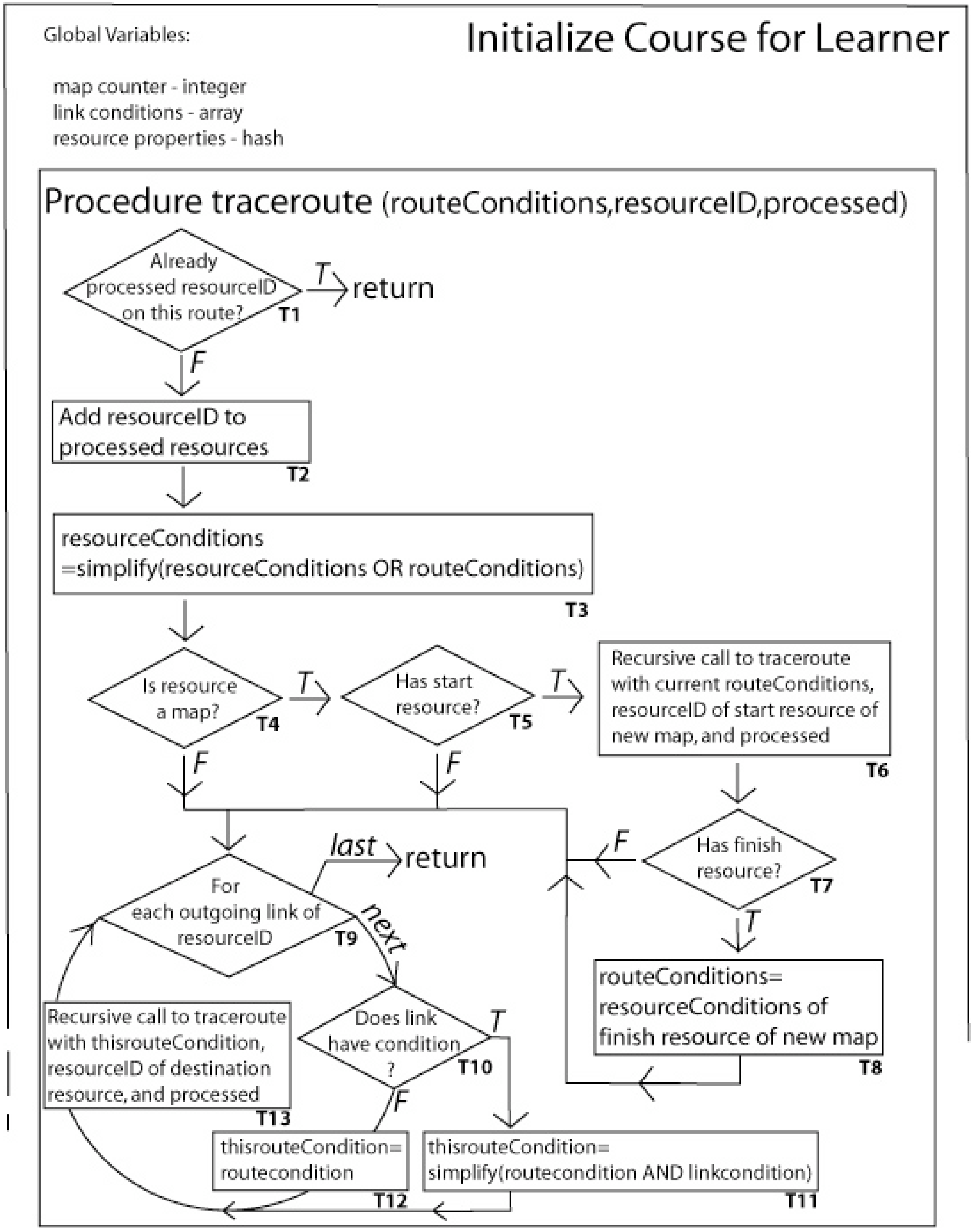

Global Variables:
Initialize Course for Learner
map counter - integer
link conditions - array
resource properties - hash
Procedure traceroute (routeConditions,resourceID,processed)
Already processed resourceID on this route?
T1
T
return
F
Add resourceID to processed resources
T2
resourceConditions =simplify(resourceConditions OR routeConditions)
T3
Is resource a map?
T4
Has start resource?
T5
Recursive call to traceroute with current routeConditions, resourceID of start resource of new map, and processed
T6
Has finish resource?
T7
routeConditions= resourceConditions of finish resource of new map
T8
For each outgoing link of resourceID
T9
last
next
Does link have condition ?
T10
thisrouteCondition= simplify(routecondition AND linkcondition)
T11
thisrouteCondition= routecondition
T12
Recursive call to traceroute with thisrouteCondition, resourceID of destination resource, and processed
T13

Fig. 2.3.8A – Flow chart of procedure `traceroute`

```
conditions_1.1: 0
conditions_1.11:
(((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7
&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8
&6&3)|(8&6&4)|(8&6&1))
conditions_1.15:
((((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7
&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8
&6&3)|(8&6&4)|(8&6&1)))&9
conditions_1.2:
(((((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&
7&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(
8&6&3)|(8&6&4)|(8&6&1)))&9)&11
conditions_1.20: 0
conditions_1.29:
((((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7
&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8
&6&3)|(8&6&4)|(8&6&1)))&9
conditions_1.36:
((((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7
&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8
&6&3)|(8&6&4)|(8&6&1)))&10
conditions_1.5: 0
conditions_1.9: 8
conditions_2.1: 8
conditions_2.12: 8
conditions_2.13: 8
conditions_2.19: 8
conditions_2.2:
(((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7
&4)|(8&7&1)))|((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8
&6&3)|(8&6&4)|(8&6&1))
conditions_2.24: ((8&5)|(8&7)|(8&6))
conditions_2.5: 8
conditions_2.6: 8
conditions_3.1: ((8&5)|(8&7)|(8&6))
conditions_3.15: ((8&5&4)|(8&7&4)|(8&6&4))
conditions_3.2:
((8&5&2)|(8&5&3)|(8&5&4)|(8&5&1)|(8&7&2)|(8&7&3)|(8&7&4)|(8&7&1)|(8&6&2)|(8&6&3)|(8&6&4
)|(8&6&1))
conditions_3.5: ((8&5)|(8&7)|(8&6))
conditions_3.6: ((8&5&2)|(8&7&2)|(8&6&2))
conditions_3.8: ((8&5&3)|(8&7&3)|(8&6&3))
```

Fig. 2.3.8B – Dump of resource properties hash. Excerpt of cumulative link conditions to reach a certain resource.

Multivalued Boolean Evaluation of Link Priorities

When a user accessed a resource on a map and desires to access the "next" resource, the request is processed by a number of steps. From the data examplified in **Fig. 2.3.6C**, it is determined which outgoing links exist. From the data in **Fig. 2.3.6D** it is determined to which resources those links lead. For each of the resources, the expressions in **Fig. 2.3.8A** are evaluated as follows. Stored in the session environment is the evaluation of the table **Fig. 10**, where the boolean part is evaluated as "0" or "1". In addition, a multivalued boolean value is computed incorporating the condition priority. A false blocking condition is assigned the value zero, all other false conditions the value 1. A true forced condition is assigned the value 3, all other true conditions the value 2.

In the expressions **Fig. 2.3.8A** an "&" ("AND") is processed as the minimum (min) operation, a "|" ("OR") is processed as the maximum (max) operation. The outcome "0" means "blocked", the outcome "1" mean "not recommended", the outcome "2" recommended, and the outcome "3" forced.

Session Four: XML Handler (Simple tags, Globals, Multiple Targets, Style Files) (Guy)

XML Files

All HTML / XML files are run through the lonxml handler before being served to a user. This allows us to rewrite many portion of a document and to support serverside tags. There are 2 ways to add new tags to the xml parsing engine, either through LON-CAPA style files or by writing Perl tag handlers for the desired tags.

Global Variables

* *$Apache::lonxml::debug* - debugging control
* *@Apache::lonxml::pwd* - path to the directory containing the file currently being processed
* *@Apache::lonxml::outputstack*
$Apache::lonxml::redirection - these two are used for capturing a subset of the output for later processing, don't touch them directly use &startredirection and &endredirection
* *$Apache::lonxml::import* - controls whether the <import> tag actually does anything
* *@Apache::lonxml::extlinks* - a list of URLs that the user is allowed to look at because of the current resource (images, and links)
* *$Apache::lonxml::metamode* - some output is turned off, the meta target wants a specific subset, use <output> to guarentee that the catianed data will be in the parsing output
* *$Apache::lonxml::evaluate* - controls whether run::evaluate actually derefences variable references
* *%Apache::lonxml::insertlist* - data structure for edit mode, determines what tags can go into what other tags
* *@Apache::lonxml::namespace* - stores the list of tag namespaces used in the insertlist.tab file that are currently active, used only in edit mode.
* *$Apache::lonxml::registered* - set to 1 once the remote has been updated to know what resource we are looking at.
* *$Apache::lonxml::request* - current Apache request object, or undef
* *$Apache::lonxml::curdepth* - current depth of the overall parse depth. Will be a string like: 2_3_1 (first tag in the third second level tag in the second toplevel tag). It gets set by callsub, and can be used in Perl tag implementations. It relies upon the internal globals: *@Apache::lonxml::depthcounter*, *$Apache::lonxml::depth*, *$Apache::lonxml::olddepth*
* *$Apache::lonxml::prevent_entity_encode* - By default the xmlparser will try to rencode any 8-bit characters into HTMLEntity Codes, If this is set to a true value it will be prevented.

In common usage, *$Apache::lonxml::prevent_entity_encode*, *$Apache::lonxml::evaluate*, *$Apache::lonxml::metamode*, *$Apache::lonxml::import*, should never be set to a value directly, but rather incremented when you want the effect on, and decremented when you want the effect off.

Notable Perl subroutines

If not specified these functions are in Apache::lonxml

* *xmlparse* - see the XMLPARSE figure - also not callable from inside a tag, if one needs to restart parsing, either create add a new LCParser to the parser stack parser using the newparser function, or call inner_xmlparser, see the xmlparse function in scripttag.pm
* *recurse* - acts just like *xmlparse*, except it doesn't do the style definition check it always calls *callsub*
* *callsub* - callsub looks if a perl subroutine is defined for the current tag and calls. Otherwise it just returns the tag as it was read in. It also will throw on a default editing interface unless the tag has a defined subroutine that either returns something or requests that call sub not add the editing interface.
* *afterburn* - called on the output of xmlparse, it can add highlights, anchors, and links to regular expersion matches to the output.
* *register_insert* - builds the %Apache::lonxml::insertlist structure of what tags can have what other tags inside.
* *whichuser* - returns a list of $symb, $courseid, $domain, $name that is correct for calls to lonnet functions for this setup. Uses form.grade_ parameters, if the user is allowed to mgr in the course
* *setup_globals* - initializes all lonxml globals when xmlparse is called. If you intend to create a new target you will likely need to tweak how the globals are setup upon start up.
* *init_safespace* - creates Holes to external functions, creates some global variables, and set the permitted operators of the global Safespace intepreter.

Functions Tag Handlers can use

If not specified these functions are in Apache::lonxml

* *debug* - a function to call to printout debugging messages. Will only print when Apache::lonxml::debug is set to 1
* *warning* - a function to use for warning messages. The message will appear at the top of a resource when it is viewed in construction space only.
* *error* - a function to use for error messages. The message will appear at the top of a resource when it is viewed in construction space, and will message the resource author and course instructor, while informing the student that an error has occured otherwise.
* *get_all_text* - 2 args, tag to look for (need to use /tag to look for an end tag) and a HTML::TokeParser reference, it will repedelyt get text from the TokeParser until the requested tag is found. It will return all of the document it pulled form the TokeParser. (See Apache::scripttag::start_script for an example of usage.)
* *get_param* - 4 arguments, first is a scaler sting of the argument needed, second is a reference to the parser arguments stack, third is a reference to the Safe space, and fourth is an optional "context" value. This subroutine allows a tag to get a tag argument, after being interpolated inside the Safe space. This should be used if the tag might use a safe space variable reference for the tag argument. (See Apache::scripttag::start_script for an example.) This version only handles scalar variables.
* *get_param_var* - 4 arguments, first is a scaler sting of the argument needed, second is a reference to the parser arguments stack, third is a reference to the Safe space, and fourth is an optional "context" value. This subroutine allows a tag to get a tag argument, after being interpolated inside the Safe space. This should be used if the tag might use a safe space variable reference for the tag argument. (See Apache::scripttag::start_script for an example.) This version can handle list or hash variables properly.
* *description* - 1 argument, the token object. This will return the textual decription of the current tag from the insertlist.tab file.
* *whichuser* - 0 arguments. This will take a look at the current environment setting and return the current $symb, $courseid, $udom, $uname. You should always use this function if you want to determine who the current user is. (Since a instructor might be trying to view a students version of a resource.)
* *inner_xmlparse* - 6 arguments, the target, an array pointer to the current stack of tags, and array pointer to the current stack of tag arguments, an array pointer to the current stack of LCParser's, a pointer to the current Safe space, a pointer to the hash of current style definitions

* *newparser* - 3 args, first is a reference to the parser stack, second should be a reference to a string scaler containg the text the newparser should run over, third should be a scaler of the directory path the file the parser is parsing was in. (See Apache::scripttag::start_import for an example.)
* *register* - should be called in a file's BEGIN block. 2 arguments, a scaler string, and a list of strings. This allows a file to register what tags it handles, and what the namespace of those tags are. Example:

```
sub BEGIN {
  &Apache::lonxml::register('Apache::scripttag',('script','display'));
}
```

Would tell xmlparse that in Apache::scripttag it can find handlers for <script> and <display>, if one regsiters a tag that was already registered the previous one is remembered and will be restored on a deregister.
* *deregister* - used to remove a previously registered tag implementation. It will restore the previous registration if there was one.
* *startredirection* - used when a tag wants to save a portion of the document for its end tag to use, but wants the intervening document to be normally processed. (See Apache::scripttag::start_window for an example.)
* *endredirection* - used to stop preventing xmlparse from hiding output. The return value is everthing that xmlparse has processed since the corresponding startredirection. (See Apache::scripttag::end_window for an example.)
* *Apache::run::evaluate* - 3 args, first a string, second a reference to the Safe space, 3 a string to be evaluated before the first arg. This subroutine will do variable interpolation and simple function interpolations on the first argument. (See Apache::lonxml::inner_xmlparse for an example.)
* *Apache::run::run* - 2 args, first a string, second a reference to the Safe space. This handles passing the passed string into the Safe space for evaluation and then returns the result. (See Apache::scripttag::start_script for an example.)

Style Files

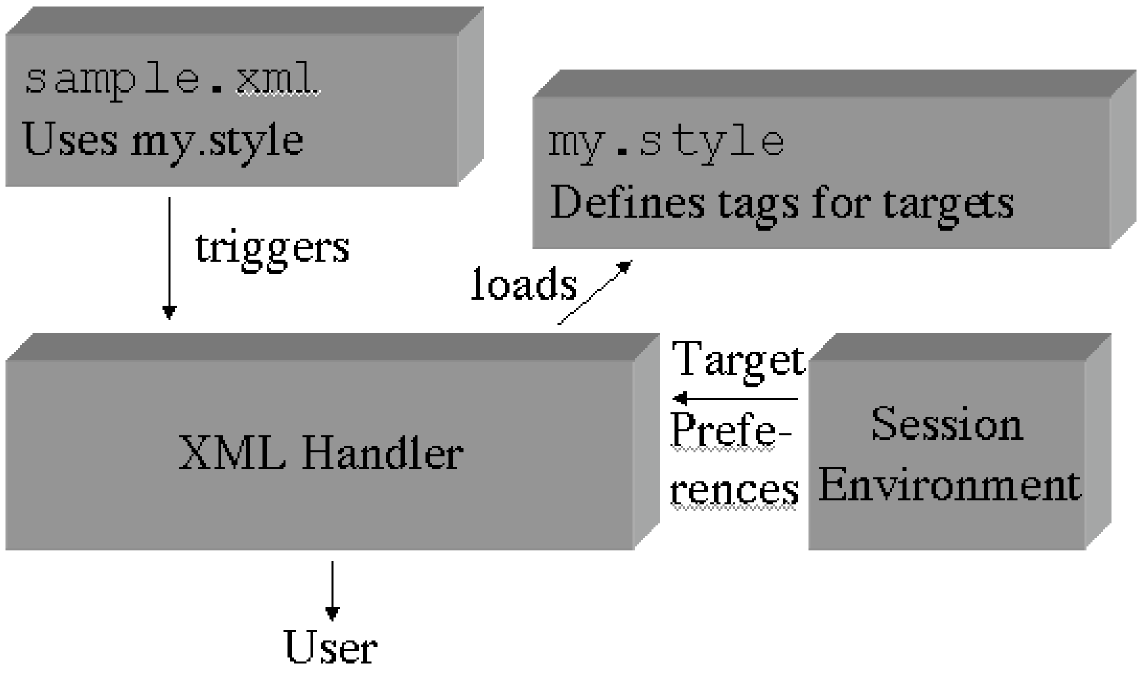


Fig. 2.4.1 – Using a style file

Style File specific tags

<definetag> - 2 arguments, *name* name of new tag being defined, if proceeded with a / defining an end tag, required; *parms* parameters of the new tag, the value of these parameters can be accesed by $parametername.
* **<render>** - define what the new tag does for a non meta target
* **<meta>** - define what the new tag does for a meta target
* **<tex>** / **<web>** / **<latexsource>** - define what a new tag does for a specific no meta target, all data inside a <render> is render to all targets except when surrounded by a specific target tags.

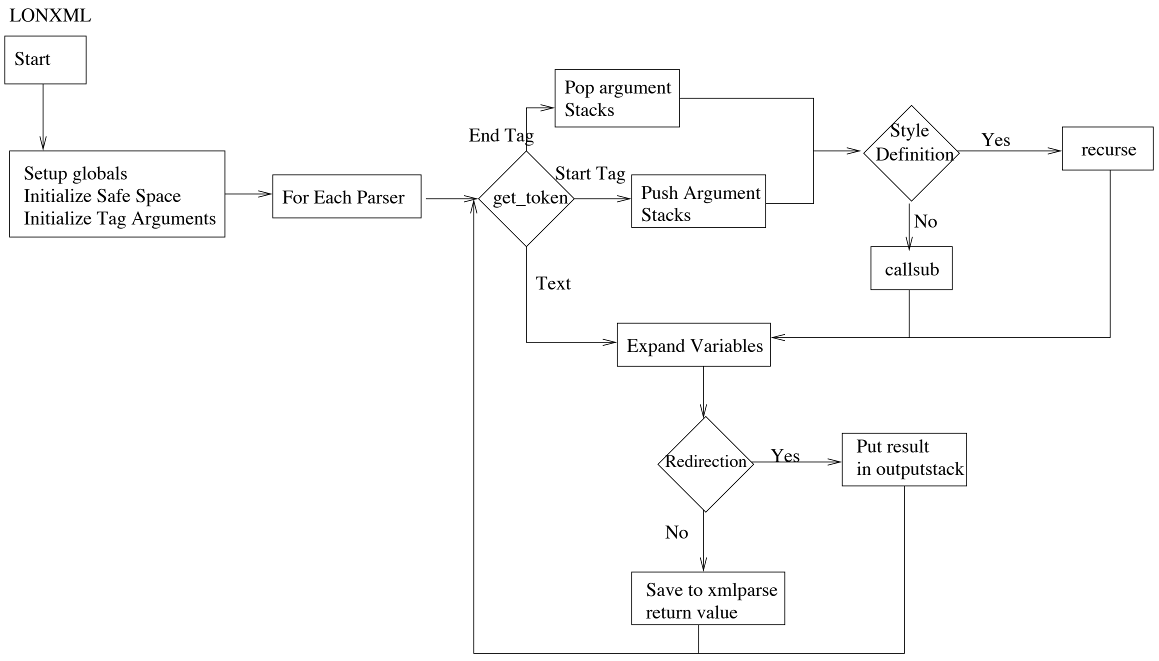


Fig. 2.4.2 – The parser

HTML::LCParser - Alternative HTML::Parser interface

```
SYNOPSIS

 require HTML::LCParser;
 $p = HTML::LCParser->new("index.html") || die "Can't open: $!";
 while (my $token = $p->get_token) {
     #...
 }

DESCRIPTION

The C<HTML::LCParser> is an alternative interface to the
C<HTML::Parser> class.  It is an C<HTML::PullParser> subclass.

The following methods are available:
```

* $p = HTML::LCParser->new($file_or_doc);

The object constructor argument is either a file name, a file handle object, or the complete document to be parsed.

If the argument is a plain scalar, then it is taken as the name of a file to be opened and parsed. If the file can't be opened for reading, then the constructor will return an undefined value and $! will tell you why it failed.

If the argument is a reference to a plain scalar, then this scalar is taken to be the literal document to parse. The value of this scalar should not be changed before all tokens have been extracted.

Otherwise the argument is taken to be some object that the C<HTML::LCParser> can read() from when it needs more data. Typically it will be a filehandle of some kind. The stream will be read() until EOF, but not closed.

It also will turn attr_encoded on by default.

* $p->get_token

This method will return the next I<token> found in the HTML document, or C<undef> at the end of the document. The token is returned as an array reference. The first element of the array will be a (mostly) single character string denoting the type of this token: "S" for start tag, "E" for end tag, "T" for text, "C" for comment, "D" for declaration, and "PI" for process instructions. The rest of the array is the same as the arguments passed to the corresponding HTML::Parser v2 compatible callbacks (see L<HTML::Parser>). In summary, returned tokens look like this:

```
  ["S",  $tag, $attr, $attrseq, $text, $line]
  ["E",  $tag, $text, $line]
  ["T",  $text, $is_data, $line]
  ["C",  $text, $line]
  ["D",  $text, $line]
  ["PI", $token0, $text, $line]
```

where $attr is a hash reference, $attrseq is an array reference and the rest are plain scalars.

* $p->unget_token($token,...)

If you find out you have read too many tokens you can push them back, so that they are returned the next time $p->get_token is called.

* $p->get_tag([$tag, ...])

This method returns the next start or end tag (skipping any other tokens), or C<undef> if there are no more tags in the document. If one or more arguments are given, then we skip tokens until one of the specified tag types is found. For example:

```
   $p->get_tag("font", "/font");
```

will find the next start or end tag for a font-element.

The tag information is returned as an array reference in the same form as for $p->get_token above, but the type code (first element) is missing. A start tag will be returned like this:

```
  [$tag, $attr, $attrseq, $text]
```

The tagname of end tags are prefixed with "/", i.e. end tag is returned like this:

```
  ["/$tag", $text]
```

* $p->get_text([$endtag])

This method returns all text found at the current position. It will return a zero length string if the next token is not text. The optional $endtag argument specifies that any text occurring before the given tag is to be returned. All entities are unmodified.

The $p->{textify} attribute is a hash that defines how certain tags can be treated as text. If the name of a start tag matches a key in this hash then this tag is converted to text. The hash value is used to specify which tag attribute to obtain the text from. If this tag attribute is missing, then the upper case name of the tag enclosed in brackets is returned, e.g. "[IMG]". The hash value can also be a subroutine reference. In this case the routine is called with the start tag token content as its argument and the return value is treated as the text.

The default $p->{textify} value is:

```
  {img => "alt", applet => "alt"}
```

This means that <IMG> and <APPLET> tags are treated as text, and that the text to substitute can be found in the ALT attribute.

* $p->get_trimmed_text([$endtag])

Same as $p->get_text above, but will collapse any sequences of white space to a single space character. Leading and trailing white space is removed.

EXAMPLES

This example extracts all links from a document. It will print one line for each link, containing the URL and the textual description between the <A>...</A> tags:

```
  use HTML::LCParser;
  $p = HTML::LCParser->new(shift||"index.html");

  while (my $token = $p->get_tag("a")) {
      my $url = $token->[1]{href} || "-";
      my $text = $p->get_trimmed_text("/a");
      print "$url\t$text\n";
  }
```

This example extract the <TITLE> from the document:

```
  use HTML::LCParser;
  $p = HTML::LCParser->new(shift||"index.html");
  if ($p->get_tag("title")) {
      my $title = $p->get_trimmed_text;
      print "Title: $title\n";
  }
```

Session Five: Worktime (Guy)

Day 3

Session One: Problem Engine/Special Targets (grade, edit, print) (Guy)

Tags

- **Response tags**

  Arguments for all response tags

  - *ID*, if this isn't set it will be set during the publication step. It is used to assign parameters names in a way that can be tracked if an instructor modifies things by hand.
  - *name* optional, if set, it will be used by the resource assembly tool when one is modifying parameters.

  Implemented response tags

  - **<responseparam>** if it appears it should be inside of a <*response> tag, defines an externally adjustable parameter for this question. Arguments:
    - *default* required, specifies a default value for the parameter
    - *name* required, specifies an internal name for the parameter
    - *type* required specifies the type of parameter, one of "tolerance", "int", "float", "string", "date" (configuration of paramters is handled by lonparmset.pm and parameter.html)
    - *description* a string describing the parameter, this is what is used to talk about a parameter outside of a problem
  - **<parameter>** exactly the same as <responseparam> currently, but should not appear inside of a <*response>
  - **<numericalresponse>** implements a numerical answer, it needs an internal **<textline>** for the response to go in. It checks all styles of numerical supported in CAPA. Possible args are:
    - *answer* required, specifies the correct answer, may be either a perl list or scalar
    - *units* optional, specifies unit of correct answer, CAPA style
  - **<stringresponse>** implements a string answer, it needs an internal **<textline>** for the response to go in. It can check the string for either case or order.
    - *answer* required, specifies the correct answer, may be either a perl list or scalar
    - *type* optional, CAPA style str args, cs/ci/mc
      - cs - case senesitive, order important
      - ci - case insenesitive, order important
      - mc - case insenesitive, order unimportant
  - **<essayresponse>** implements a ungraded large text response, it need an internal **<textarea>** for the response to go in.
  - **<imageresponse>** implements a image click style image submission, uses the foil structure tags below. Additional tags that should appear in a <foil> are:

    - **<image>** required, the contained text specifies a published graphical resource that is the image used, should only appear once per foil
    - **<rectangle>** required, the contained text specifies a rectangular area that is correct, should look like (1,2)-(3,4), at least 1 required
    - **<text>** required, the contained text is printed on top of the image.
- **<optionresponse>** implements a "select from these choices" style question, the choices are specified by the instructor, it uses the foil structure tags below with this additional args:
    - **<foilgroup>** is required to have *options* which should be a perl list of possible options for the student.
- **<radiobuttonresponse>** implements a true / false style question with 1 correct answer.it uses the foil structure tags below but the *value* of a <foil>can only be "true" or "false" or "unused"
- **<dataresponse>** implements a straight data storage entry idea, needs and interveing input tag like <textline> to work correctly.
  Arguments:
    - *name* internal name for the value, it will have the part id and respose id added on to it
    - *type* type of data stored in this response field, should be one of the types supported by parameter.html
    - *display* a string that will be used to describe the field when interfacing with humans
- **<externalresponse>** implements the ability to have an external program grade a response, expects either a <textline> or <textfield> inside the tag.

  Arguments:
    - *url* the url to submit the answer and form to, does not need to be a LON-CAPA machine.
    - *answer* a string or scalar variable that can encode something that should encode the correct answer, in some cases this may be nothing.
    - *form* a hash variable name that will be submitted to the remote site as a HTTP form.

  The response of the remote server needs to be in XML as follows.

  ```
   <loncapagrade>     <awardetail> CORRECT     </awardetail>
  <message> A message to be shown to the students
  </message> </loncapagrade>
  ```
    - **<loncapagrade>** no arguments but must surround the response.
    - **<awardetail>** required inner tag, the response inside must be one of the detailed responses that appears in the data storage documentation (CVS:loncapa/doce/homework/datastorage)
    - **<message>** optional message to have shown to the student

- **Foil Structure Tags**

  All tags that implement a foil structure have an optional arg of *max* that controls the maximum number of total foils to show.

  - **<foilgroup>** required, must be the tag that surrounds all foil definitions
  - **<foil>** required, all data inside is a possible foil
  - **<conceptgroup>** optional, surrounds a collection of <foil>, when a problem is displayed only one of the contained <foil>is selected for display. It receives one required argument *concept*.

- **Hint structure**

  All of these tags must appear inside a **<*response>** tag.

  - **<hintgroup>** Tag that surrounds all of a hint.
  - **<hintpart>** required, Tag to implement conditional hints. It has a required argument *on*. When a <*hint> tag named the same as the value the on attribute evaluates to be correct the <hintpart> will show. If no other <hintpart> are to show then all hintparts with a *on* of "default" will show
  - **<numericalhint>** has all the arguments that <numericalresponse>, does and the required attribute *name* which should be set to the value of which <hintpart> will be shown.

- **Input Tags**

  This group of tags implement a mechanism for getting data for students, they will usually be used by a <*response>.

  - **<textarea>** creates a Large text input box, If data appears between the start and end tags, the data will appear i the textarea if the student has not yet made a submission. Additionally it takes two arguments *rows* and *cols* which control the height and width of the area respectively. It defaults to 10 and 80.
  - **<textline>** creates a single line of input element, it accepts 1 argument *size* which controls the width on the textline, it defaults to 20.

- **Output Tags**

  This group of tags generate useful pieces of output.

  - **<standalone>** everything in between the start and end tag is shown only on the web, and only if the resource is not part of a course.
  - **<displayduedate>** this will insert the current duedate if one is set into the document. It is generated to be inside a table of 1x1 elements
  - **<displaytitle>** this will insert the title of the problem from the metadata of the problem
  - **<window>** the text in between is put in a popup javascript window

- **<m>** the inside text is LaTeX, and is converted to HTML (or MathML) on the fly, if the argument *eval* is set to "on" the intervening text will have a perl var expansion done to it before being converted.

- **<randomlabel>** shows a specified image with images or text labels randomly assigned to a set of specific locations, those locations may also have values assigned to them. There is a hash generated conating the mapping of labels to locations, labels to values, and locations to values. Example:

  ```
    <randomlabel bgimg="URL" width="12" height="45"
  texwidth="50">     <labelgroup name="GroupOne"
  type="image">       <location x="123" y="456" value="10" />
  <location x="321" y="654" value="20" />       <location
  x="213" y="546" value="13" />       <label
  description="TEXT-1">IMG-URL</label>       <label
  description="TEXT-2">IMG-URL</label>       <label
  description="TEXT-3">IMG-URL</label>     </labelgroup>
  <labelgroup name="GroupTwo" type="text">       <location
  x="12" y="45" />       <location x="32" y="65" />
  <location x="21" y="54" />       <label>TEXT-1</label>
  <label>TEXT-2</label>       <label>TEXT-3</label>
  </labelgroup>    </randomlabel>
  ```
  Arguments:

  - *bgimg* either a fully qualified URL for an external image, or a loncapa resource, it supports relative references (../images/apicture.gif), the image must either be a GIF or JPEG
  - *width* the width of the image in pixels
  - *height* the height of the image in pixels
  - *texwidth* the width of the image in millimeters

  Internal tags:

  - **<labelgroup>** 1 required, multiple allowed. Declares a group of locations and labels associated with them.
    Arguments:
    - *name* this is the name of the group, a hash with this name will be generated holding the mappings for later use in the problem. For each location a value will be set for which label is there, (EX. $hash{'1'}="TEXT-2"). For locations with values the hash will contain 2 items, a location to value mapping ($hash{'value_1'}=10), and a label to value mapping ($hash{'labelvalue_2'}=10). For all image style of labels there will also be a label description to label URL mapping ($hash{'image_2'}=IMG-URL). Also the entry 'numlocations will be set to the total number of locations that exist. (Note that locations and labels start counting from 1.)
    - *type* the type of labels in this group, either 'image' or 'text'
  - **<location>** declares a location on the image that a label should appear at
    Arguments:
    - *x* the x value of the location in pixels
    - *y* the y value of the location in pixels
    - *value* a scalar value to associate at this location (optional)

        - **<label>** declaration of a label, if this is a text type labelgroup the internal text should be the text of the label (HTML is not currently supported), if this is an image type of label the internal text must be a LON-CAPA resource specification, and the description filed must be set.
          Arguments:
            - *description* a required field for image labels, it will be used when setting values in the hash.

- **Scripting**

  These tags allow the document to behave programatically

    - **<display>** the intervening perl script is evaluated in the safe space and the return value of the script replaces the entire tag
    - **<import>** causes the parse to read in the file named in the body of the tag and parse it as if the entire text of the file had existed at location of the tag
    - **<parserlib>** the enclosed filename contains definitions for new tags
    - **<script>** if the argument *type* is set to "loncapa/perl" the enclosed data is a perl script which is evaluated inside the perl Safe space. The return value of the script is ignored.
    - **<scriptlib>** the enclosed filename contains perl code to run in the safe space
    - **<block>** has a required argument *condition* that is evaluated, it the condition is true everything inside the tag is evaluated, if it is false everything inside the block tag is skipped
    - **<notsolved>** everything inside the tag is skipped if the problem is "solved"
    - **<postanswerdate>** everything inside the tag is skipped if the problem is before the answer date
    - **<preduedate>** everything inside the tag is skipped if the problem is after the due date
    - **<randomlist>** the enclosed tags are parsed in a stable random order, optional argument *show* restricts the number of tags indie that are actually parsed the no more than *show*.
    - **<solved>** everything inside the tag is skipped if the problem is "not solved"
    - **<while>** implements a while loop, required argument *condition* is a perl scriptlet that when evaluated results in a true or false value, on true the entirety of the text between the whiles is parsed. The condition is tested again, etc. If false it goes to the next node in the parse.

- **Structure Tags**

  These tags give the problem a structure and take care of the recording of data and giving the student messages.

    - **<problem>** must be the first tag in the file, this tag sets up the header of the webpage and generates the submit buttons, it also handles due dates properly
    - **<part>** must be below <problem> if it is going to be used. It does many of the same tasks as <problem> but allows multiple separate problems to exist in a single file.

- **<startouttext><endouttext>** these tags are somewhat special, they must have no internal text and occur in pairs. Their use is to mark up the problem so the web editor knows what sections should be edited in a plain text block on the web.

<script> Functions

A list of functions that have been written that are available in the Safe space scripting environment inside a problem.

- sin(x), cos(x), tan(x)
- asin(x), acos(x), atan(x), atan2(y,x)
- log(x), log10(x)
- exp(), pow(x,y), sqrt(x)
- abs(x), sgn(x)
- erf(x), erfc(x)
- ceil(x), floor(x)
- min(...), max(...)
- factorial(n)
- N%M
- sinh(x), cosh(x), tanh(x)
- asinh(x), acosh(x), atanh(x)
- roundto(x,n)
- web("a","b","c") or web(a,b,c)
- html("a") or html(a)
- j0(x), j1(x), jn(n,x), jv(y,x)
- y0(x), y1(x), yn(n,x), yv(y,x)
- random
- choose
- tex("a","b") or tex(a,b)
- var_in_tex(a)
- to_string(x), to_string(x,y)
- class(), section()
- name(), student_number()
- open_date(), due_date(), answer_date()

- sub_string()
- array_moments(array)
- format(x,y),prettyprint(x,y)
- map(...)
- caparesponse_check
- caparesponse_check_list

Detailed descriptions of each function and comparison with CAPA

CAPA Functions	**LON-CAPA**	**Descriptions**	**Differences (if any)**
sin(x), cos(x), tan(x)	&sin($x), &cos($x), &tan($x)	Trigonometric functions where x is in radians. $x can be a pure number, i.e., you can call &sin(3.1415)	
asin(x), acos(x), atan(x), atan2(y,x)	&asin($x), &acos($x), &atan($x), &atan2($y,$x)	Inverse trigonometric functions. Return value is in radians. For asin and acos the value of x must be between -1 and 1. The atan2 returns a value between -pi and pi the sign of which is determined by y. $x and $y can be pure numbers	
log(x), log10(x)	&log($x), &log10($x)	Natural and base-10 logarithm. $x can be a pure number	
exp(x), pow(x,y), sqrt(x)	&exp($x), &pow($x,$y), &sqrt($x)	Exponential, power and square root, i.e.,e^x, x^y and /x. $x and $y can be pure numbers	
abs(x), sgn(x)	&abs($x), &sgn($x)	Abs takes the absolute value of x while sgn(x) returns 1, 0 or -1 depending on the value of x. For x>0, sgn(x) = 1, for x=0, sgn(x) = 0 and for x<0, sgn(x) = -1. $x can be a pure number	
erf(x), erfc(x)	&erf($x), &erfc($x)	Error function. erf = 2/sqrt(pi) integral (0,x) e^{t-sq} and *erfx(x)* = 1.0 - *erf(x)*. $x can be a pure number	
ceil(x), floor(x)	&ceil($x), &floor($x)	Ceil function returns an integer rounded up whereas floor function returns and integer rounded down. If x is an integer than it returns the value of the integer. $x can	

		be a pure number	
min(...), max(...)	&min(...), &max(...)	Returns the minimum/ maximum value of a list of arguments if the arguments are numbers. If the arguments are strings then it returns a string sorted according to the ASCII codes	
factorial(n)	&factorial($n)	Argument (n) must be an integer else it will round down. The largest value for n is 170. $n can be a pure number	
N%M	$N%$M	N and M are integers and returns the remainder (in integer) of N/M. $N and $M can be pure numbers	
sinh(x), cosh(x), tanh(x)	&sinh($x), &cosh($x), &tanh($x)	Hyperbolic functions. $x can be a pure number	
asinh(x), acosh(x), atanh(x)	&asinh($x), &acosh($x), &atanh($x)	Inverse hyperbolic functions. $x can be a pure number	
/DIS($x,"nn")	&format($x,"nn")	Display or format $x as nn where nn is nF or nE and n is an integer.	The difference is obvious.
Not in CAPA	&prettyprint($x,"nn")	Display or format $x as nn where nn is nF or nE and n is an integer. In E mode it will attempt to generate a pretty x10^3 rather than a E3 following the number	
roundto(x,n)	&roundto($x,$n)	Rounds a real number to n decimal points. $x and $n can be pure numbers	
web("a","b","c") or web(a,b,c)	&web("a","b","c") or &web($a,$b,$c)	Returns either a, b or c depending on the output medium. a is for plain ASCII, b for tex output and c for html output	
html("a") or html(a)	&html("a") or &html($a)	Output only if the output mode chosen is in html format	
jn(m,x)	&j0($x), &j1($x), &jn($m,$x), &jv($y,$x)	Bessel functions of the first kind with orders 0, 1 and m respectively. For jn(m,x), m must be an integer whereas for jv(y,x), y is real. $x can be a pure number. $m must be an integer and can be a pure integer number. $y can be a pure real number	In CAPA, j0, j1 and jn are contained in one function, jn(m,x) where m takes the value of 0, 1 or 2. jv(y,x) is new to LON-CAPA.
yn(m,x)	&y0($x), &y1($x),	Bessel functions of the	In CAPA, y0, y1

	&yn($m,$x), &yv($y,$x)	second kind with orders 0, 1 and m respectively. For yn(m,x), m must be an integer whereas for yv(y,x), y is real. $x can be a pure number. $m must be an integer and can be a pure integer number. $y can be a pure real number	and yn are contained in one function, yn(m,x) where m takes the value of 0, 1 or 2. yv(y,x) is new to LON-CAPA.
random(l,u,d)	&random($l,$u,$d)	Returns a uniformly distributed random number between the lower bound, l and upper bound, u in steps of d. $l, $u and $d can be pure numbers	In CAPA, all the 3 arguments must be of the same type. However, now you can mix the type
choose(i,...)	&choose($i,...)	Choose the ith item from the argument list. i must be an integer greater than 0 and the value of i should not exceed the number of items. $i can be a pure integer	
/MAP(seed;w,x,y,z;a,b,c,d)	Option 1 - &map($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &map($seed,\@mappedArray,[$a,$b,$c,$d]) Option 3 - @mappedArray = &map($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &map($seed,\@a) where $a='A' $b='B' $c='B' $d='B' $w, $x, $y, and $z are variables	Assigns to the variables $w, $x, $y and $z the values of the $a, $b, $c and $c (A, B, C and D). The precise value for $w .. depends on the seed. (Option 1 of calling map). In option 2, the values of $a, $b .. are mapped into the array, @mappedArray. The two options illustrate the different grouping. Options 3 and 4 give a consistent way (with other functions) of mapping the items. For each option, the group can be passed as an array, for example, [$a,$b,$c,$d] => \@a.	In CAPA, the arguments are divided into three groups separated by a semicolon ;. In LON-CAPA, the separation is done by using [] brackets or using an array @a. Note the backslash (\) before the arguments in the second and third groups.
rmap(seed;a,b,c,d;w,x,y,z)	Option 1 - &rmap($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &rmap($seed,\@rmappedArray,[$a,$b,$c,$d]) Option 3 - @rmapped_array = &rmap($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &rmap($seed,\@a) where $a='A' $b='B' $c='B'	The rmap functions does the reverse action of map if the same seed is used in calling map and rmap.	In CAPA, the arguments are divided into three groups separated by a semicolon ;. In LON-CAPA, the separation is done by using [] brackets (with create an unamed vector reference) or using an array @a. Note the backslash (\) before the arguments in the

	$d='B' $w, $x, $y, and $z are variables		second and third groups (Which cause Perl to send to variable locations rather than the variable values, similar to a C pointer).
NOT IMPLEMENTED IN CAPA	$a=&xmlparse($string)	Runs the internal parser over the argument parsing for display. **Warning** This will result in different strings in different targets. Don't use the results of this function as an answer.	New to LON-CAPA
tex(a,b), tex("a","b")	&tex($a,$b), &tex("a","b")	Returns a if the output mode is in tex otherwise returns b	
var_in_tex(a)	&var_in_tex($a)	Equivalent to tex("a","")	
to_string(x), to_string(x,y)	&to_string($x), &to_string($x,$y)	If x is an integer, returns a string. If x is real than the output is a string with format given by y. For example, if x = 12.3456, &to_string(x,".3F") = 12.345 and &to_string(x,".3E") = 1.234E+01.	
capa_id(), class(), section(), set(), problem()	&class(), &section()	Returns null string, class descriptive name, section number, set number and null string.	capa_id(), set() and problem() are no longer used. Currently, they return a null value.
name(), student_number()	&name(), &student_number()	Return the full name in the following format: lastname, firstname initial. Student_number returns the student 9-alphanumeric string. If undefined, the functions return null.	
open_date(), due_date(), answer_date()	&open_date(), &due_date(), &answer_date()	Problem open date, due date and answer date. The time is also included in 24-hr format.	Output format for time is changed slightly. If pass noon, it displays ..pm else it displays ..am. So 23:59 is displayed as 11:59 pm.
get_seed(), set_seed()	Not implemented	Get and set the random seed.	
sub_string(a,b,c)	&sub_string($a,$b,$c) perl substr function. However, note the differences	Retrieve a portion of string a starting from b and length c. For example, $a = "Welcome to LON-CAPA"; $result=&sub_string($a,4,4);	Perl intrinsic function, substr(string,b,c) starts counting from 0 (as opposed

		then $result is "come"	to 1). In the example to the left, substr($a,4,4) returns "ome ".
array[xx]	@arrayname Array is intrinsic in perl. To access a specific element use $arrayname[$n] where $n is the $n+1 element since the array count starts from 0	"xx" can be a variable or a calculation.	In LON-CAPA, an array is defined by @arrayname. It is not necessary to specify the dimension of the array.
array_moments(B,A)	@B=&array_moments(@A)	Evaluates the moments of an array A and place the result in array B[i] where i = 0 to 4. The contents of B are as follows: B[0] = number of elements, B[1] = mean, B[2] = variance, B[3] = skewness and B[4] = kurtosis.	In CAPA, the moments are passed as an array in the first argument whereas in LON-CAPA, the array containing the moments are set equal to the function.
array_max(Name), array_min(Name)	&min(@Name), &max(@Name)	In LON-CAPA to find the maximum value of an array, use &max(@arrayname) and to find the minimum value of an array, use &min(@arrayname)	Combined with the min and max functions defined earlier.
init_array(Name)	undef @name	To destroy the contents of an array, use	Use perl intrinsic undef function.
random_normal (return_array,item_cnt,seed,av,std_dev)	@return_array=&random_normal ($item_cnt,$seed,$av,$std_dev)	Generate $item_cnt deviates of normal distribution of average $av and standard deviation $std_dev. The distribution is generated from seed $seed	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_beta (return_array,item_cnt,seed,aa,bb)	@return_array=&random_beta ($item_cnt,$seed,$aa,$bb) NOTE: Both $aa and $bb MUST be greater than 1.0E-37.	Generate $item_cnt deviates of beta distribution. The density of beta is: X^($aa-1) *(1-X)^($bb-1) /B($aa,$bb) for 0<X<1.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_gamma (return_array,item_cnt,seed,a,r)	@return_array=&random_gamma ($item_cnt,$seed,$a,$r) NOTE: Both $a and $r MUST be positive.	Generate $item_cnt deviates of gamma distribution. The density of gamma is: ($a**$r)/gamma($r) * X**($r-1) * exp(-$a*X).	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_exponential	@return_array=&random	Generate $item_cnt deviates	In CAPA the

(return_array,item_cnt,seed,av)	exponential ($item_cnt,$seed,$av) NOTE: $av MUST be non-negative.	of exponential distribution.	results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_poisson (return_array,item_cnt,seed,mu)	@return_array=&random_poisson ($item_cnt,$seed,$mu) NOTE: $mu MUST be non-negative.	Generate $item_cnt deviates of poisson distribution.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_chi (return_array,item_cnt,seed,df)	@return_array=&random_chi ($item_cnt,$seed,$df) NOTE: $df MUST be positive.	Generate $item_cnt deviates of chi_square distribution with $df degrees of freedom.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_noncentral_chi (return_array,item_cnt,seed,df,nonc)	@return_array=&random_noncentral_chi ($item_cnt,$seed,$df,$nonc) NOTE: $df MUST be at least 1 and $nonc MUST be non-negative.	Generate $item_cnt deviates of noncentral_chi_square distribution with $df degrees of freedom and noncentrality parameter $nonc.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
NOT IMPLEMENTED IN CAPA	@return_array=&random_f ($item_cnt,$seed,$dfn,$dfd) NOTE: Both $dfn and $dfd MUST be positive.	Generate $item_cnt deviates of F (variance ratio) distribution with degrees of freedom $dfn (numerator) and $dfd (denominator).	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_noncentral_f ($item_cnt,$seed,$dfn,$dfd,$nonc) NOTE: $dfn must be at least 1, $dfd MUST be positive, and $nonc must be non-negative.	Generate $item_cnt deviates of noncentral F (variance ratio) distribution with degrees of freedom $dfn (numerator) and $dfd (denominator). $nonc is the noncentrality parameter.	New to LON-CAPA
NOT DOCUMENTED IN CAPA	@return_array=&random_multivariate_normal ($item_cnt,$seed,\@mean,\@covar) NOTE: @mean should be of length p array of real numbers. @covar should be a length p array of references to length p arrays of real numbers (i.e. a p by p matrix.	Generate $item_cnt deviates of multivariate_normal distribution with mean vector @mean and variance-covariance matrix.	Note the backslash before the @mean and @covar arrays.
NOT IMPLEMENTED	@return_array=&random	Returns single observation	New to LON-

IN CAPA	multinomial ($item_cnt,$seed,@p) NOTE: $item_cnt is rounded with int() and the result must be non-negative. The number of elements in @p must be at least 2.	from multinomial distribution with $item_cnt events classified into as many categories as the length of @p. The probability of an event being classified into category i is given by ith element of @p. The observation is an array with length equal to @p, so when called in a scalar context it returns the length of @p. The sum of the elements of the obervation is equal to $item_cnt.	CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_ permutation ($item_cnt,@array)	Returns @array randomly permuted.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_ uniform ($item_cnt,$seed,$low,$high) NOTE: $low must be less than or equal to $high.	Generate $item_cnt deviates from a uniform distribution.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_ uniform_integer ($item_cnt,$seed,$low,$high) NOTE: $low and $high are both passed through int(). $low must be less than or equal to $high.	Generate $item_cnt deviates from a uniform distribution in integers.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_ binomial ($item_cnt,$seed,$nt,$p) NOTE: $nt is rounded using int() and the result must be non-negative. $p must be between 0 and 1 inclusive.	Generate $item_cnt deviates from the binomial distribution with $nt trials and the probabilty of an event in each trial is $p.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_ negative_binomial ($item_cnt,$seed,$ne,$p) NOTE: $ne is rounded using int() and the result must be positive. $p must be between 0 and 1 exclusive.	Generate an array of $item_cnt outcomes generated from negative binomial distribution with $ne events and the probabilty of an event in each trial is $p.	New to LON-CAPA

<script> Variables

- $external::target - set to the current target the xml parser is parsing for
- $external::part - set to the *id* of the current problem <part>; zero if there are no <part>

- $external::gradestatus - set to the value of the current resource.partid.solved value
- $external::datestatus - set to the current status of the clock either CLOSED, CAN_ANSWER, CANNOT_ANSWER, SHOW_ANSWER, or UNCHECKEDOUT
- $external::randomseed - set to the number that was used to seed the random number generator
- $pi - set to PI
- $rad2deg - converts radians to degrees
- $deg2rad - converts degrees to radians

Form internals

Form elements used for homework editing/response

the form is named lonhomework

Viewing a problem (either CSTR or RES space)

- grade_target, grade_username, grade_domain, grade_symb - the vaules take precedence over the the normal ENV settings of these, use &Apache::lonxml::whichuser() to get back the correct vaues of the items
- HWVAL_response:number - the name of the input elements for each response. The response is the response's ID, and the :number part exists only for response with multiple foils, and is a unique number in order of the submission
- changerandseed - the 'Change' button, only exists right after the Change button is clicked
- problemmode - in CSTR, it is either 'View','Edit','EditXML' or 'Analyze' depending on the desired view
- rndseed - the currently requested random seed
- showallfoils - if it exists, and we are in CSTR space, the problem should ignore <conceptgroup> and the response's max setting and show all possible foils
- submit - the name of the Submit button
- submitted - a hidden form parameter that can be used to tell if the student has submitted answers or not, if it is set, there should be answers to grade.

Editing a problem (Edit mode)

tagdepth referes to the current value of the xmlparsers tagdepth couter ($Apache::lonxml::currentdepth)

Note: the file edit.pm has many helper functions for creating the standard elements for editing a tag and it's attributes, all of those functions follow these conventions

- tagdepth_argname - I.E. 1_3.max, a parameter editing value, max is the name of the parameter and it is for the third tag inside the first tag
- delete_tagdepth - if set to yes, delete tag tagdepth and all tags inside of it

- homework_edit_tagdepth - used by both &Apache::edit::editline and &Apache::edit::editfield for the <textarea> and <input> form elements they create
- insert_tagdepth - used to request an tag insert, it is set to the id number of the requested tag in the %Apache::lonxml::insertlist
-

Symbs

To identify a specific instance of a resource, LON-CAPA uses symbols or "symbs." These identifiers are built from the URL of the map, the resource number of the resource in the map, and the URL of the resource itself. The latter is somewhat redundant, but might help if maps change.

An example is

```
msu/korte/parts/part1.sequence___19___msu/korte/tests/part12.problem
```

The respective map entry is

```
<resource id="19" src="/res/msu/korte/tests/part12.problem"
 title="Problem 2">
</resource>
```

Symbs are used by the random number generator, as well as to store and restore data specific to a certain instance of for example a problem.

Store / Restore

2 important functions in lonnet.pm are `&Apache::lonnet::cstore()` and `&Apache::lonnet:restore()` (and `&Apache::lonnet::store()`, which is is the non-critical message twin of `cstore`). These functions are for handlers to store a perl hash to a user's permanent data space in an easy manner, and to retrieve it again on another call. It is expected that a handler would use this once at the beginning to retrieve data, and then again once at the end to send only the new data back.

The data is stored in the user's data directory on the user's homeserver under the ID of the course.

The hash that is returned by `restore` will have all of the previous value for all of the elements of the hash.

Example:

```
#creating a hash
my %hash;
$hash{'foo'}='bar';
#storing it
&Apache::lonnet::cstore(\%hash);
#changing a value
$hash{'foo'}='notbar';
#adding a new value
$hash{'bar'}='foo';
&Apache::lonnet::cstore(\%hash);
#retrieving the hash
my %history=&Apache::lonnet::restore();
#print the hash
foreach my $key (sort(keys(%history))) {
    print("\%history{$key} = $history{$key}");
}
```

Will print out:

```
%history{1:foo} = bar
%history{1:keys} = foo:timestamp
%history{1:timestamp} = 990455579
%history{2:bar} = foo
%history{2:foo} = notbar
%history{2:keys} = foo:bar:timestamp
%history{2:timestamp} = 990455580
%history{bar} = foo
%history{foo} = notbar
%history{timestamp} = 990455580
%history{version} = 2
```

Note that the special hash entries *keys*, *version* and *timestamp* were added to the hash. *version* will be equal to the total number of versions of the data that have been stored. The *timestamp* attribute will be the UNIX time the hash was stored. *keys* is available in every historical section to list which keys were added or changed at a specific historical revision of a hash.

Warning do not store the hash that restore returns directly. This will cause a mess since it will restore the historical keys as if the were new keys. I.E. `1:foo` will become `1:1:foo` etc.

Calling convention:

```
  my %record=&Apache::lonnet::restore($symb,$courseid,$domain,$uname,$home);
  &Apache::lonnet::cstore(\%newrecord,$symb,$courseid,$domain,$uname,$home);
```

Arguments (only %newrecord is required the rest are somewhat optional, read the details):

- *$symb* - a string containing the internal name of the specific instance of a resource. Usually this value can be gotten from `&Apache::lonnet::symbread($filename)`. If the argument is blank, it will attempt to use `symbread()` for it. If the result is ambiguous store/restore will fail.
- *$courseid* - the internal name for a course, usually found in `$ENV{'request.course.id'}` which is what will be looked at if no value is passed to the functions.
- *$domain* - the domain that the user belongs to, usually found in `$ENV{'user.domain'}` which is what will be looked at if no value is passed to the functions.
- *$uname* - the login name for the user, usually found in `$ENV{'user.name'}` which is what will be looked at if no value is passed to the functions.
- *$home* - the homeserver for the user, usually found in `$ENV{'user.home'}` but can be easily gotten from a domain and name through `&Apache::lonnet::homeserver($uname,$domain)`. If no value is passed to store/restore the value in %ENV will be used.
- *%newrecord* - the hash to store being passed by reference

Return values:

- *an empty string* - the function was unable to determine exactly where to store or restore from. At least one of the "optional" arguments was unable to be determined.

- *a hash* - restore successfully read a old hash for this specific user / resource instance.
- *no_such_host* - the *$home* specfied desn't exist in the network.
- *con_delayed* - the *$home* was uncontactable at this time. The store will be delayed until it is again available.
- *con_failed* - the *$home* was uncontactable at this time and store was unable to delay the store until a later time. The store failed.
- *ok* - the store completed succesfully
- *error:* - remote server failied to store or restore the reason follows the **:**

Mandatory Homework Data

```
<Provided by &EXT() and set external to the resource, required>
resource.partid.opendate   #unix time of when the local machine should let the
                           #student in

resource.partid.duedate    #unix time of when the local machine should stop
                           #accepting answers

resource.partid.answerdate #unix time of when the local machine should
                           #provide the correct answer to the student

resource.partid.weight     # points the problem is worth

resource.partid.maxtries   # maximum number of attempts the student can have

resource.partid.type       # type of problem homework can be:
                           # homework - randomized, graded, stored with
                           #            requesting user, full feeback
                           # exam - randomized, graded, stored with
                           #        requesting user, minimal feedback
                           # form - unrandomized, ungraded, stored with
                           #        specified user, full feedback
                           # survey - unrandomized, ungraded, stored with
                           #          requesting user, full feedback



<numerical/formula/response needed: (has a default if nonexistant)>

resource.partid.responseid.tol   # lots of possibilities here
                    # percentage, range (inclusive and exclusive),
                    # variable name, etc
                    # 3%
                    # 0.5
                    # .05+
                    # 3%+
                    # 0.5+,.005

resource.partid.responseid.sig  # one or two comma sepearted integers,
                                # specifying the number of significatn figures
                                # a student must use



<Problem sets using cstore (required):>
resource.partid.solved # if not set, problem yet to be viewed
                # incorrect_attempted == incorrect and attempted
                # correct_by_student == correct by student work
                # correct_by_override == correct, instructor override
                # incorrect_by_override == incorrect, instructor override
                # excused == excused, problem no longer counts for student
                # '' (empty) == not attempted
                # ungraded_attempted == an ungraded answer has been
                                        sumbitted and stored
resource.partid.tries  # positive integer of number of unsuccessful attempts
                # made, malformed answers don't count if feedback is
                # on

resource.partid.awarded # float between 0 and 1, percentage of
                 # resource.weight that the stundent earned.

resource.partid.award # final detailed award that was applied to the entire
                      # part of the question, check awarddetail below for
                      # possibilities

resource.partid.previous # boolean, is this submission a previous submission

resource.partid.responseid.submissons
                    # the student submitted string for the part.response

resource.partid.responseid.awarddetail
                     # list of all of the results of grading the submissions
                     # in detailed form of the specific failure
                     #Possible values:
                     # EXACT_ANS, APPROX_ANS : student is correct
                     # NO_RESPONSE : student submitted no response
                     # MISSING_ANSWER : student submitted some but not
                     #                   all parts of a response
                     # WANTED_NUMERIC : expected a numeric answer and
                     #                   didn't get one
                     # SIG_FAIL : incorrect number of Significant Figures
                     # UNIT_FAIL : incorrect unit
                     # UNIT_NOTNEEDED : Submitted a unit when one shouldn't
                     # NO_UNIT : needed a unit but none was submitted
                     # BAD_FORMULA : syntax error in submitted formula
                     # INCORRECT : answer was wrong
                     # SUBMITTED : submission wasn't graded
                     # ERROR : unable to get a grade

resource.partid.responseid.message (optional) (not yet supported)
                      # a message that should be shown to the student

resource.partid.bonustries (optional) (not yet supported)
                     # if set, added to the maxtries parameter for student
                     # total number of tries overall
```

Sample Problems

A Simple Problem

```
<problem>
        <script type="loncapa/perl">
$length=&random(10,99,.1);
$width=&random(1,10,.01);
@area=($length*($width*10));
        </script>

What is the area of a box $length mm in length and
```

```
&format($width,"2E") cm in width.


        <numericalresponse id="11" answer="@area" units="mm^2">

                <textline></textline>

                <responseparam name="tol" type="tolerance" default="5%"></responseparam>

        </numericalresponse>


</problem>
```

A More Complex Problem

```
<problem>

<displayduedate />
<p><displaytitle /></p>
<script type="loncapa/perl">
$vF="<b> F<sub>1</sub> </b>";
$vF1="<b> F<sub>1</sub> </b>";
$vF2="<b> F<sub>2</sub> </b>";

$mF="|<b>F</b>|";
$F1mag="|<b>F<sub>1</sub></b>|";
$F2mag="|<b>F<sub>2</sub></b>|";
$trq1mag="|<b> <font face=symbol>t</font><sub>1</sub></b>|";
$trq2mag="|<b> <font face=symbol>t</font><sub>2</sub></b>|";
$Q1="Q<sub>1</sub>";
$Q2="Q<sub>2</sub>";
$tau="<font face=symbol>t</font>";
$tau1="<font face=symbol><b>t</b></font><sub>1</sub>";
   $val=&random(1,4,1);
   $tp=&choose($val,"her","her","his","his");
   $sd=&choose($val,"daughter","niece","nephew","son");

</script>
<startouttext />
$trq1mag and $trq2mag are the magnitudes of the torques produced repectively by
forces $vF1 and $vF2 with respect the pivot P. The magnitudes of $vF1 is
$F1mag and that of $vF2 is $F2mag. $Q1 and $Q2 are the locations
on a rigid body where $vF1 and $vF2 act.

<endouttext />
<optionresponse max="600">
        <foilgroup options="('Correct','Incorrect','Can not tell')">
                <conceptgroup concept="Effect of the moment-arm on the torque">
                        <foil name="1a" value="Incorrect">
                          For $F1mag larger than $F2mag , $trq1mag is larger than
$trq2mag
                        </foil>
                        <foil name="1b" value="Incorrect">
                          For $F1mag smaller than $F2mag , $trq1mag is smaller than
$trq2mag
                        </foil>
                        <foil name="1c" value="Correct">
                          For $F1mag larger than $F2mag , $trq1mag can be less  than
$trq2mag
                        </foil>
                        <foil name="1d" value="Correct">
                          For $F1mag smaller than $F2mag , $trq1mag can be larger  than
$trq2mag
                        </foil>
                </conceptgroup>
                <conceptgroup concept="For a given pivot, relation of force vector to
torque.">
                        <foil name="2a" value="Correct">
                  The moment-arm of $vF is the shortest distance from P to the line along
$vF .
                        </foil>
                        <foil name="2b" value="Incorrect">
                          The moment-arm of $vF is the shortest distance from P to $vF
vector.
                        </foil>
                        <foil name="2c" value="Correct">
                          The moment-arm of $vF is not the distance from P to
Q<sub>1</sub>.
                        </foil>
                        <foil name="2d" value="Incorrect">
                          The moment-arm of $vF is the distance from P to Q<sub>1</sub>.
                        </foil>
                </conceptgroup>
                <conceptgroup concept="Torque is force x moment-arm.">
                        <foil name="3a" value="Correct">
                          $trq1mag equals to the product of the moment-arm and $F1mag .
                      </foil>
                        <foil name="3b" value="Incorrect">
                        $trq1mag equals to $F1mag times the distance from Q<sub>1</sub>
to P.
                        </foil>
                        <foil name="3c" value="Correct">
                          $trq1mag is not equal to the product of $F1mag and the distance
from Q<sub>1</sub> to P.
                        </foil>
                        <foil name="3d" value="Incorrect">
                        $trq1mag is not equal to the product of the moment-arm and $F1mag
.
                        </foil>
                </conceptgroup>
                <conceptgroup concept="Pivot point is required to calculate torque">
                        <foil name="4a" value="Correct">
                         $tau1 vector depends on the location of P.
                        </foil>
                        <foil name="4b" value="Incorrect">
                         $tau1 vector does not depend on the location of P.
                        </foil>
                        <foil name="4c" value="Correct">
                         $tau1 vector has no meaning unless a pivot is selected.
                        </foil>
                        <foil name="4d" value="Incorrect">
                         $tau1 vector can be determined without selecting P.
                        </foil>
                </conceptgroup>
                <conceptgroup concept="torque from 2 forces acting along same line">
                        <foil name="5a" value="Correct">
                          Two equal forces, acting on a body along the same line but at
different positions, produce equal torques with respect to a given pivot."
                        </foil>
                        <foil name="5b" value="Incorrect">
                          Two equal forces, along the same line, produce equal torques
with respect to a given pivot only if they act at the same point on a body."
                        </foil>
                        <foil name="5c" value="Incorrect">
                          Two equal forces acting on a body along the same line but at
different positions, produce equal torques for only one pivot."
                        </foil>
                </conceptgroup>

                <foil name="6" value="unused">
                        This foil will never display since it is unused.
                </foil>
        </foilgroup>
        <notsolved>
                <hintgroup>
Think the definition of the torque. The force and the moment-arm respect to the pivot.
                </hintgroup>
        </notsolved>
</optionresponse>
</problem>
```

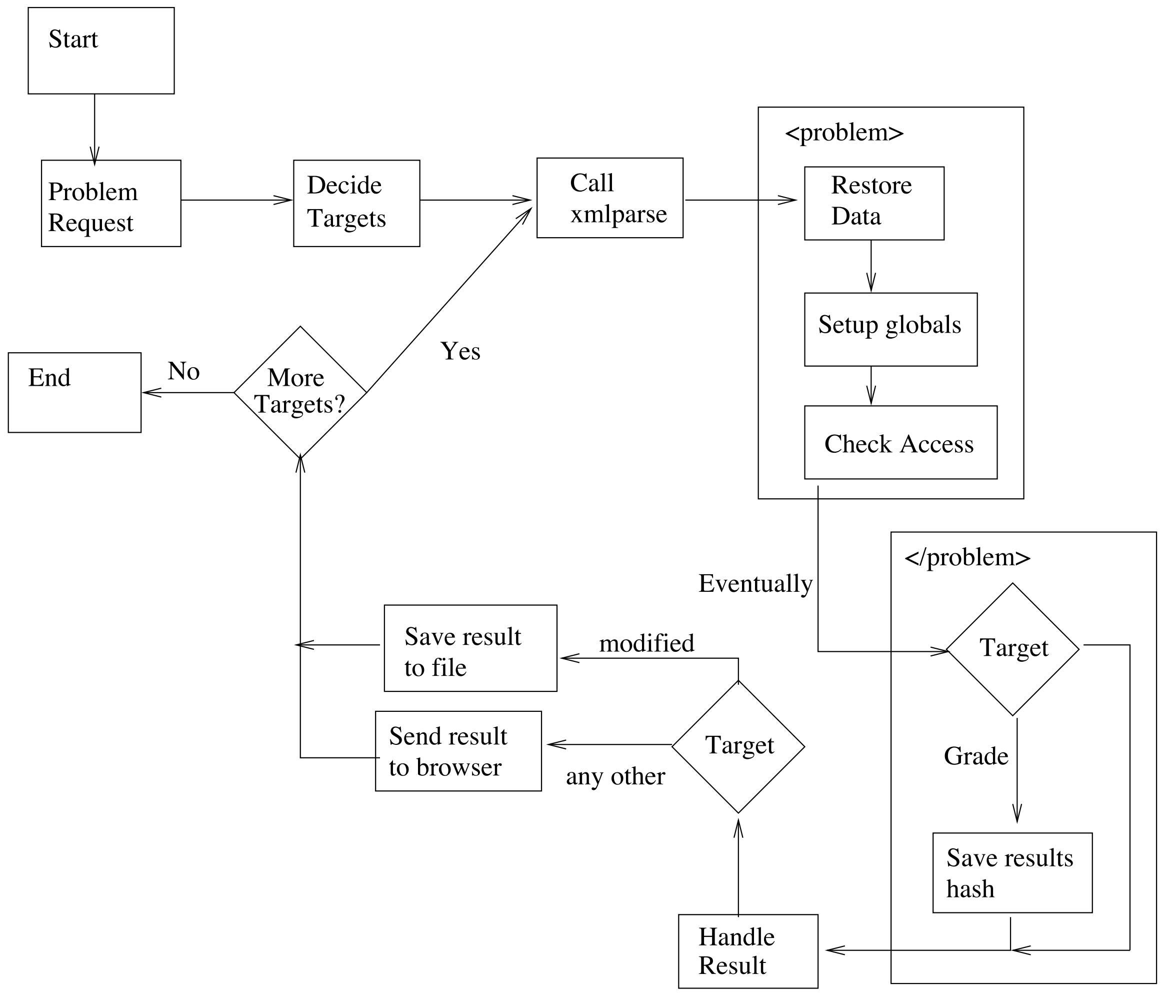


Fig. 3.1.1 – Internal Structure of Homework Handler

Session Two: Statistics/Chart (Behrouz)

In LON-CAPA, we are involved with two kinds of large data sets:

1) Educational resources such as web pages, demonstrations, simulations, and individualized problems designed for use on homework assignments, quizzes, and examinations;
2) Information about users who create, modify, assess, or use these resources.

Usually, instructors/ course coordinators wish to assess the students' educational situation or evaluate the problems have bee presented in the course, after the students used the educational materials. There are two main modules in LON_CAPA that provide the statistical information for instructors/ course-coordinators: *lonchart.pm* and *lonstatistics.pm*. When an instructor's access is authorized, he/she can find useful reports about the course, regarding maps, problems included in each map, and the students who tried to solve the problems. When an instructor selects a course, he/she has two buttons in remote control to obtain the statistical information: "*chart*" or "*stat*" button.

lonchart.pm

“*chart*” button in remote control calls lonchart.pm, which provides a quick review of students tries on different problems of a course for an instructor. The instructor may monitor the number of tries of every student in each map and its problems. The number of solved problems in a map is shown in the end of each map with a different color (green). The overall solved problems and total number of problems in the map can be seen at the end of line according to every individual student in a different color (blue). A sample of chart is shown in **Fig. 3.2.1**

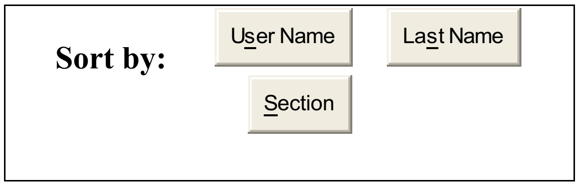
```
xxxxxxx1:msu ! 001 ! 1*11*121 8 1231.31423212 12  2111211284.131 13 …  231113112221 12  162 / 188
xxxxxxx2:msu ! 003 ! 12113162 8 1+11  1  21x11110 11211322246132 14 … ############    0  149 / 188
```

1..9: correct by student in 1..9 tries
*: correct by student in more than 9 tries
+: correct by override
-: incorrect by override
.: incorrect attempted
#: ungraded attempted
‘ ‘: not attempted
x: excused

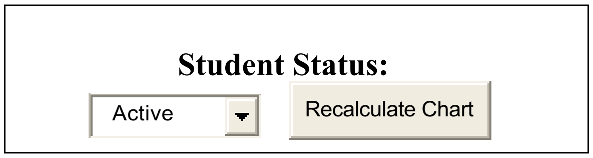
Fig. 3.2.1 Chart of map, problems, and students’ tries, and a quick statistics of solved problems

1. When an instructor loads the chart in his/her machine once, its data is cached in his/her local machine. If he/she runs the chart again, the course chart is loaded very quickly from the cache.

2. An instructor can sort the chart according to user name, last name, as well as, the section which student belongs to.



3. The instructor is able to select the “expired” students (who dropped the course earlier) or “active” students or “any” (all the) students.



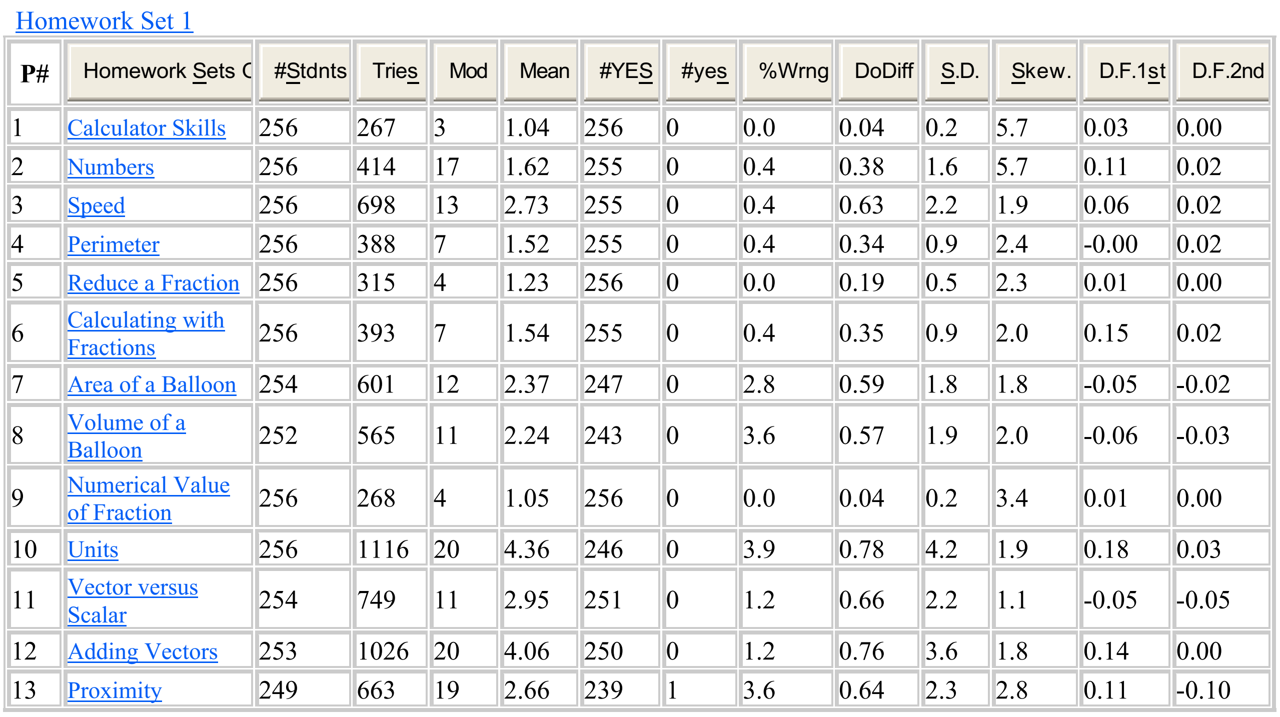
lonstatistics.pm

In “stat” button of “remote control”, a menu with three options is provided for instructor:
1) Problem stats

2) Problem Analysis
3) Student Assessment

Problem Stats

"*Problem Stats*" button provides a table[1] which includes statistical information about every problem, as you see in **Fig. 3.2.2**. The function ExtractStudentData in lonstatitics Perl Module fetches all the data from a particular student .db file into a big hash in local machine. It uses dump function, which communicates via lonc/lond to get the student data from student repository (data server) and then all versions of student submissions computed according to every problem. The results are stored in an array in the memory. Before computing the students' tries in a particular problem, the different parts of problem are distinguished by considering the meta-data, which is provided for every problem.



Homework Set 1

P#	Homework Sets C	#Stdnts	Tries	Mod	Mean	#YES	#yes	%Wrng	DoDiff	S.D.	Skew.	D.F.1st	D.F.2nd
1	Calculator Skills	256	267	3	1.04	256	0	0.0	0.04	0.2	5.7	0.03	0.00
2	Numbers	256	414	17	1.62	255	0	0.4	0.38	1.6	5.7	0.11	0.02
3	Speed	256	698	13	2.73	255	0	0.4	0.63	2.2	1.9	0.06	0.02
4	Perimeter	256	388	7	1.52	255	0	0.4	0.34	0.9	2.4	-0.00	0.02
5	Reduce a Fraction	256	315	4	1.23	256	0	0.0	0.19	0.5	2.3	0.01	0.00
6	Calculating with Fractions	256	393	7	1.54	255	0	0.4	0.35	0.9	2.0	0.15	0.02
7	Area of a Balloon	254	601	12	2.37	247	0	2.8	0.59	1.8	1.8	-0.05	-0.02
8	Volume of a Balloon	252	565	11	2.24	243	0	3.6	0.57	1.9	2.0	-0.06	-0.03
9	Numerical Value of Fraction	256	268	4	1.05	256	0	0.0	0.04	0.2	3.4	0.01	0.00
10	Units	256	1116	20	4.36	246	0	3.9	0.78	4.2	1.9	0.18	0.03
11	Vector versus Scalar	254	749	11	2.95	251	0	1.2	0.66	2.2	1.1	-0.05	-0.05
12	Adding Vectors	253	1026	20	4.06	250	0	1.2	0.76	3.6	1.8	0.14	0.00
13	Proximity	249	663	19	2.66	239	1	3.6	0.64	2.3	2.8	0.11	-0.10

Fig. 3.2.2: Statistics table includes general statistics of every problem of the course

Every part of *multi-part* problems is distinguished as a separate problem. The *multi-instance* problem is also considered separately, because a particular problem or one part of it might be used in different maps. Finally, the array, which includes all computed information from all students, sorted according to the problem order, underlying in homework sets order. Therefore, in this step we can compute the following statistical information:

1. **#Stdnts**: Total number of students who take a look at the problem.(Let #Stdnts is equal to *n*)

[1] If instructor is going to port the statistics table data to Excel, he/she can select the checkbox "Output CSV format" at top of the statistics table.

2. **Tries:** Total number of tries to solve the problem ($\sum_{i=1}^{n} x_i$ where x_i denote a student try).
3. **Mod:** Mode, Maximum Number of Tries for solving the problem.
4. **Mean:** Average Number of the Tries.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

5. **#YES:** Number of students solved the problem correctly.
6. **#yes:** Number of students solved the problem by override.
   Sometimes, a student gets a correct answer after talking with the instructor. This type of correct answer is called "corrected by override.

7. **%Wrng:** Percentage of students tried to solve the problem but still incorrect.

$$100 * \left(\frac{n - (\#YES + \#yes)}{n} \right)$$

8. **S.D.:** *Standard Deviation* of the students' tries.

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

9. **Skew.:** *Skewness* of the students' tries.

$$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{(S.D.)^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{\left(\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}\right)^3}$$

10. **DoDiff:** *Degree of Difficulty* of the problem.

$$1 - \left(\frac{\#YES + \#yes}{\sum_{i=1}^{n} x_i} \right)$$

   As you see Degree of Difficulty is always between 0 and 1. This is a good factor for an instructor to determine whether a problem is difficult, and what is the degree of this difficulty. Thus, DoDiff of each problem is saved in its meta data.

11. **Dis.F.:** *Discrimination Factor*[2] is an standard for evaluating how much a problem discriminates between the upper and the lower students. First, all of the students are sorted according to a criterion. Then, %27 of upper students and %27 lower students are selected from the sorted students applying the mentioned criterion. Finally we obtain the Discrimination Factor from the following difference:
   Applied a criterion in %27 upper students - Applied the same Criterion in %27 lower students.
   Discrimination Factor is a number in interval [-1,1]. If this number is close to 1, it shows that only upper students have solved this problem. If it is close to 0 it shows that the upper students and the lowers are approximately the same in solving the problem. If this number is negative, it shows that the lower students have more successes in solving the problem, and thus this problem is very poor in discriminating the upper and lower students.
   In lonstatistics.pm we compute the Discrimination Factor from two criteria:

[2] This name has been got from administration office of Michigan State University for evaluating the exams' problem. Here we expanded this expression to homework problems as well.

1st Criterion for Sorting the Students: $\dfrac{\sum \text{Partial Credit Awarded}}{\sum_{i=1}^{n} x_i}$

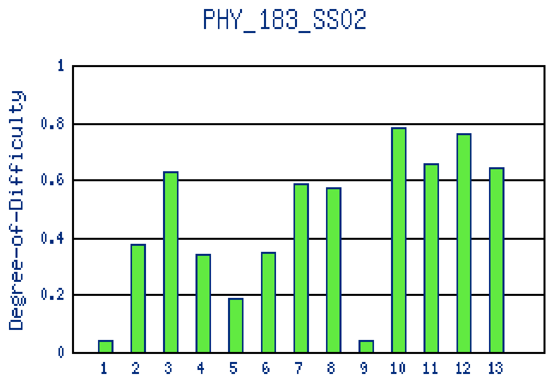
2nd Criterion for Sorting the Students: $\dfrac{\sum (\#YES + \#yes)}{\sum_{i=1}^{n} x_i}$

- **Change the stats table sorting**

As you see in Fig. 3.2. 2, all headers in the stats table are buttons that change the order of the table. Users can change increasingly or decreasingly every column of the table. First the user select the "*ascending*" or "*descending*" option, then he/she can change the order of the table with clicking the header of his/her interested header. If the user changes the order the table, all information is shown in one table, each row corresponds to a particular problem. If the user selects the first column, "homework set order", the information is shown in different tables, each table corresponds to a particular homework set.

- **Graphical chart**

Two important features in this page might be seen through the graphical charts. That is, a user could see the content of "%wrong" column and "degree of difficulty of problems" in the graphical chart as is shown in Fig. 3.2. 3 and 4 for homework set 1 in course PHY183 SS02. These graphical charts are produced dynamically by calling a CGI scripts, (graph.gif) which is located in /home/httpd/cgi-bin/



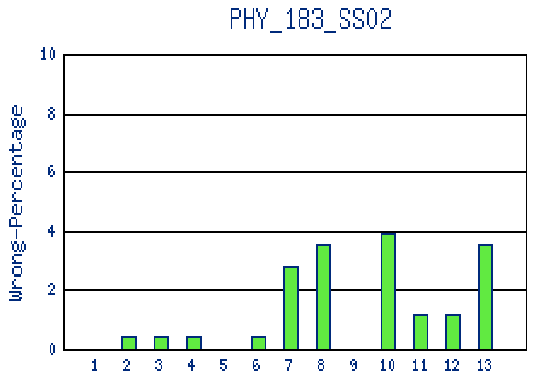
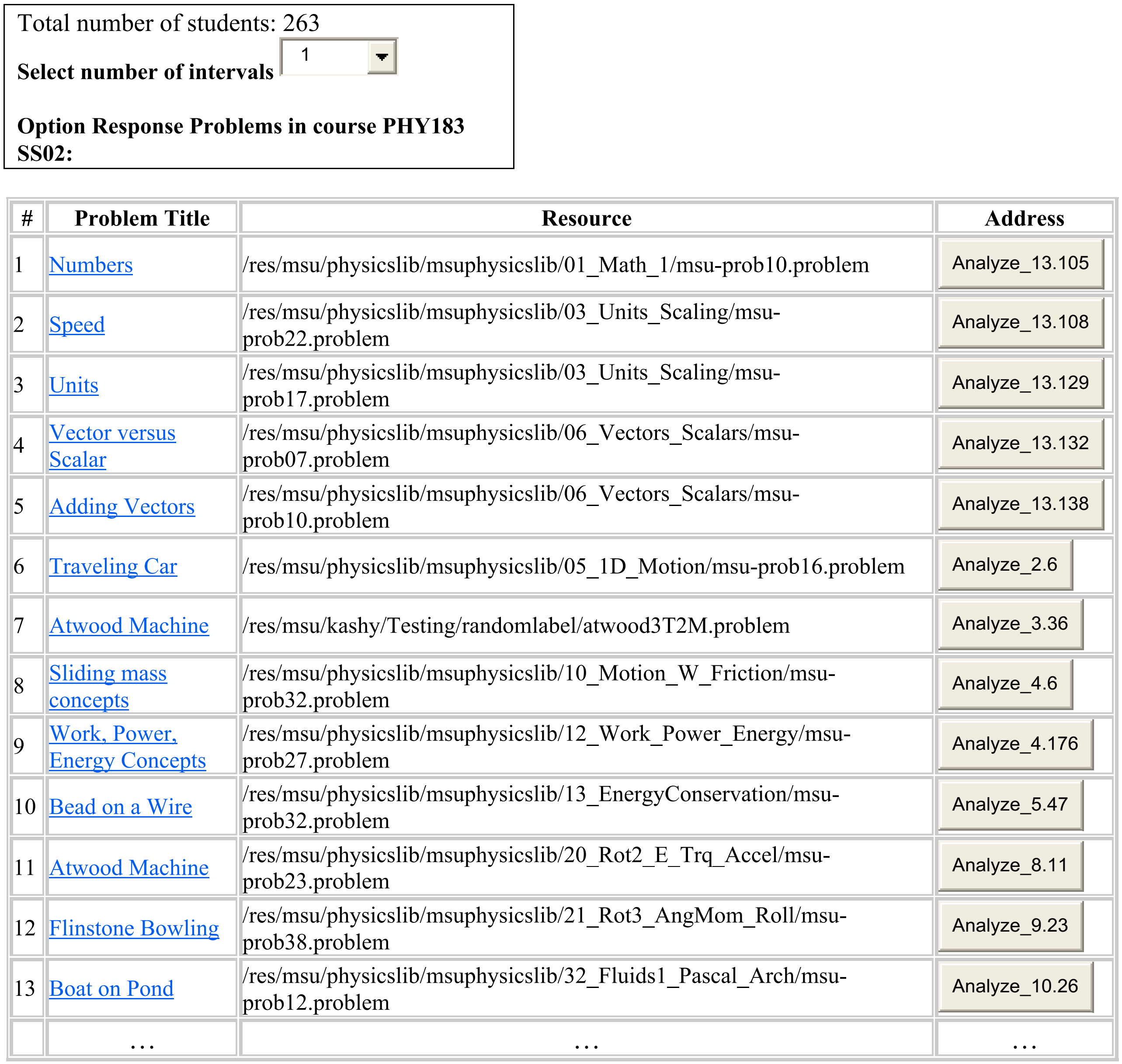


Fig. 3.2. 3: Degree of difficulty graph

Fig. 3.2. 4: %Wrong graph

Problem Analysis

Conceptual option response problems, in which the students are given several concepts that are randomly assigned to each student, are more difficult than numerical simple problems. Instructors usually want to see the students' tries according to every particular concept separately. "Problem Analysis" button provides all response option problems in one table as follows in the **Fig. 3.2.5.**



Total number of students: 263

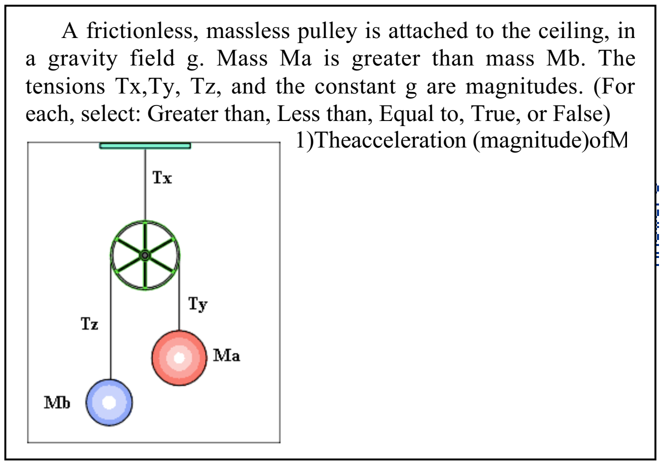
Select number of intervals 1

Option Response Problems in course PHY183 SS02:

#	Problem Title	Resource	Address
1	Numbers	/res/msu/physicslib/msuphysicslib/01_Math_1/msu-prob10.problem	Analyze_13.105
2	Speed	/res/msu/physicslib/msuphysicslib/03_Units_Scaling/msu-prob22.problem	Analyze_13.108
3	Units	/res/msu/physicslib/msuphysicslib/03_Units_Scaling/msu-prob17.problem	Analyze_13.129
4	Vector versus Scalar	/res/msu/physicslib/msuphysicslib/06_Vectors_Scalars/msu-prob07.problem	Analyze_13.132
5	Adding Vectors	/res/msu/physicslib/msuphysicslib/06_Vectors_Scalars/msu-prob10.problem	Analyze_13.138
6	Traveling Car	/res/msu/physicslib/msuphysicslib/05_1D_Motion/msu-prob16.problem	Analyze_2.6
7	Atwood Machine	/res/msu/kashy/Testing/randomlabel/atwood3T2M.problem	Analyze_3.36
8	Sliding mass concepts	/res/msu/physicslib/msuphysicslib/10_Motion_W_Friction/msu-prob32.problem	Analyze_4.6
9	Work, Power, Energy Concepts	/res/msu/physicslib/msuphysicslib/12_Work_Power_Energy/msu-prob27.problem	Analyze_4.176
10	Bead on a Wire	/res/msu/physicslib/msuphysicslib/13_EnergyConservation/msu-prob32.problem	Analyze_5.47
11	Atwood Machine	/res/msu/physicslib/msuphysicslib/20_Rot2_E_Trq_Accel/msu-prob23.problem	Analyze_8.11
12	Flinstone Bowling	/res/msu/physicslib/msuphysicslib/21_Rot3_AngMom_Roll/msu-prob38.problem	Analyze_9.23
13	Boat on Pond	/res/msu/physicslib/msuphysicslib/32_Fluids1_Pascal_Arch/msu-prob12.problem	Analyze_10.26
	…	…	…

Fig. 3.2.5: Option response problems in course PHY183 SS02

Fig. 3.2.5 includes a table, which shows the title of every option response problem in the first column. This title has a link to the original html page of the problem. In the second column the source address of the problem is shown. Third column of this table includes a button to analyze the students' data on this particular option response problem. When this button is clicked, all data about this problem is *restored*, for every student. Different versions of students' submissions are evaluated. The results are presented in a graphical chart as well as a numerical table. For example, if we select the analysis of **/res/msu/kashy/Testing/randomlabel/atwood3T2M.problem**


A frictionless, massless pulley is attached to the ceiling, in a gravity field g. Mass Ma is greater than mass Mb. The tensions Tx,Ty, Tz, and the constant g are magnitudes. (For each, select: Greater than, Less than, Equal to, True, or False)

1)Theacceleration (magnitude)ofM

Fig. 3.2.6: Atwood Machine option response problem in HW3

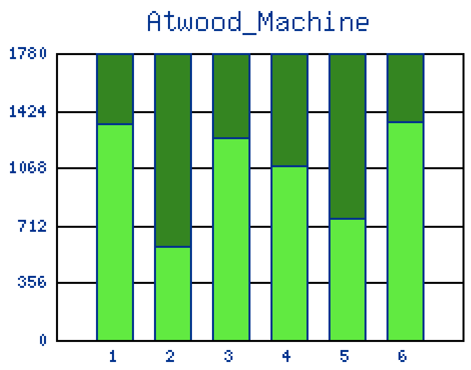
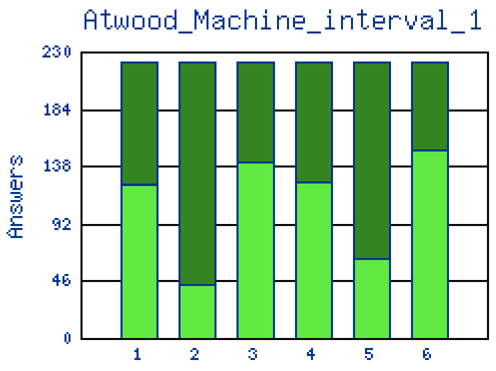


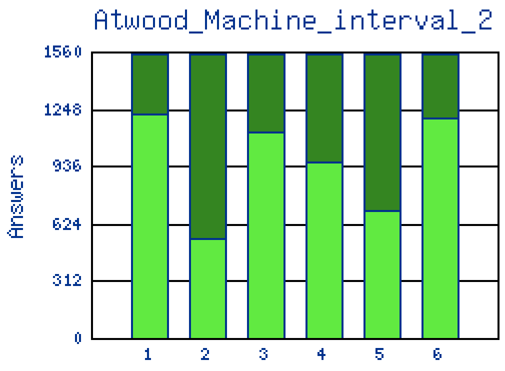
Fig. 3.2.7: Graphical chart of student tries for Atwood Machine Problem according to every concept in 1 interval time

In addition, the data of students' tries are shown in a table as you see in **Fig. 3.2.8**. In the last row of the table you can see the time interval of this data and the overall correct and wrong answers separately. If an instructor wants to see the students' tries in different time intervals, he/she could set the number of intervals from 1 to 7 time intervals, and then recompute the analysis.

#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	1342	433
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	585	1190
3	The top tension is equals the two bottom tensions. (massless pulley)	1263	512
4	Tension holding the two masses are equal if mass of pulley=0	1087	688
5	Sub-System accelerates upwards or downwards accordingly	757	1018
6	Center of mass accelerates downward	1354	421
	From:[Thu Jan 24 00:46:22 2002] To: [Mon Feb 4 23:59:59 2002]	6388	4245

Fig. 3.2.8: Table of student tries for Atwood Machine Problem according to every concept in one time interval.



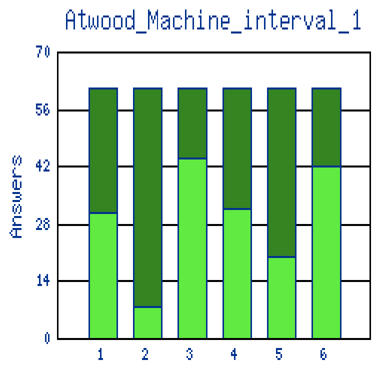


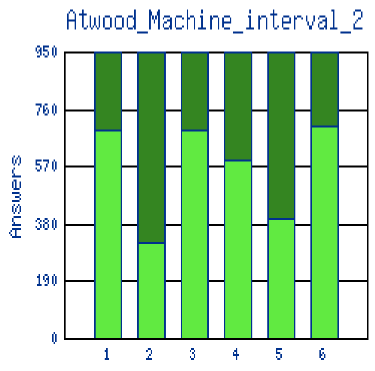
#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	124	98
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	44	178
3	The top tension is equals the two bottom tensions. (massless pulley)	142	80
4	Tension holding the two masses are equal if mass of pulley=0	125	97
5	Sub-System accelerates upwards or downwards accordingly	64	158
6	Center of mass accelerates downward	151	71
	From:[Thu Jan 24 00:46:22 2002] To: [Wed Jan 30 00:23:10 2002]	650	682

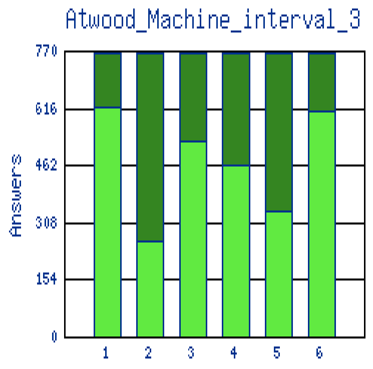
#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	1218	335
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	541	1012
3	The top tension is equals the two bottom tensions. (massless pulley)	1121	432
4	Tension holding the two masses are equal if mass of pulley=0	962	591
5	Sub-System accelerates upwards or downwards accordingly	693	860
6	Center of mass accelerates downward	1203	350
	From:[Wed Jan 30 00:23:11 2002] To: [Mon Feb 4 23:59:59 2002]	5738	3563

Fig. 3.2.9: Table of student tries for Atwood Machine Problem according to every concept in 2 times interval.

In **Fig. 3.2.9**, number of students' tries tables and the graphical chart are shown in 2 different time intervals. An instructor would be able to check whether the students have more wrong answers during the first days of opening the homework set, and how many students have tried during the first or the second interval. Since the problems are individualized he/she might be able to see how many students have tried to solve the problem after communicating with each other and understanding the concept. In Fig. 3.2. 10 the charts and tables of students' tries are shown in 3 time intervals. So if the homework should be done in one week, an instructor would be able to observe the distribution of students' tries every day separately after choosing the 7 time intervals.







#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	31	30
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	8	53
3	The top tension is equals the two bottom tensions. (massless pulley)	44	17
4	Tension holding the two masses are equal if mass of pulley=0	32	29
5	Sub-System accelerates upwards or downwards accordingly	20	41
6	Center of mass accelerates downward	42	19
	From:[Thu Jan 24 00:46:22 2002] To: [Mon Jan 28 00:30:53 2002]	177	189

#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	692	257
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	321	628
3	The top tension is equals the two bottom tensions. (massless pulley)	690	259
4	Tension holding the two masses are equal if mass of pulley=0	590	359
5	Sub-System accelerates upwards or downwards accordingly	399	550
6	Center of mass accelerates downward	703	246
	From:[Mon Jan 28 00:30:54 2002] To: [Fri Feb 1 00:15:25 2002]	3395	2281

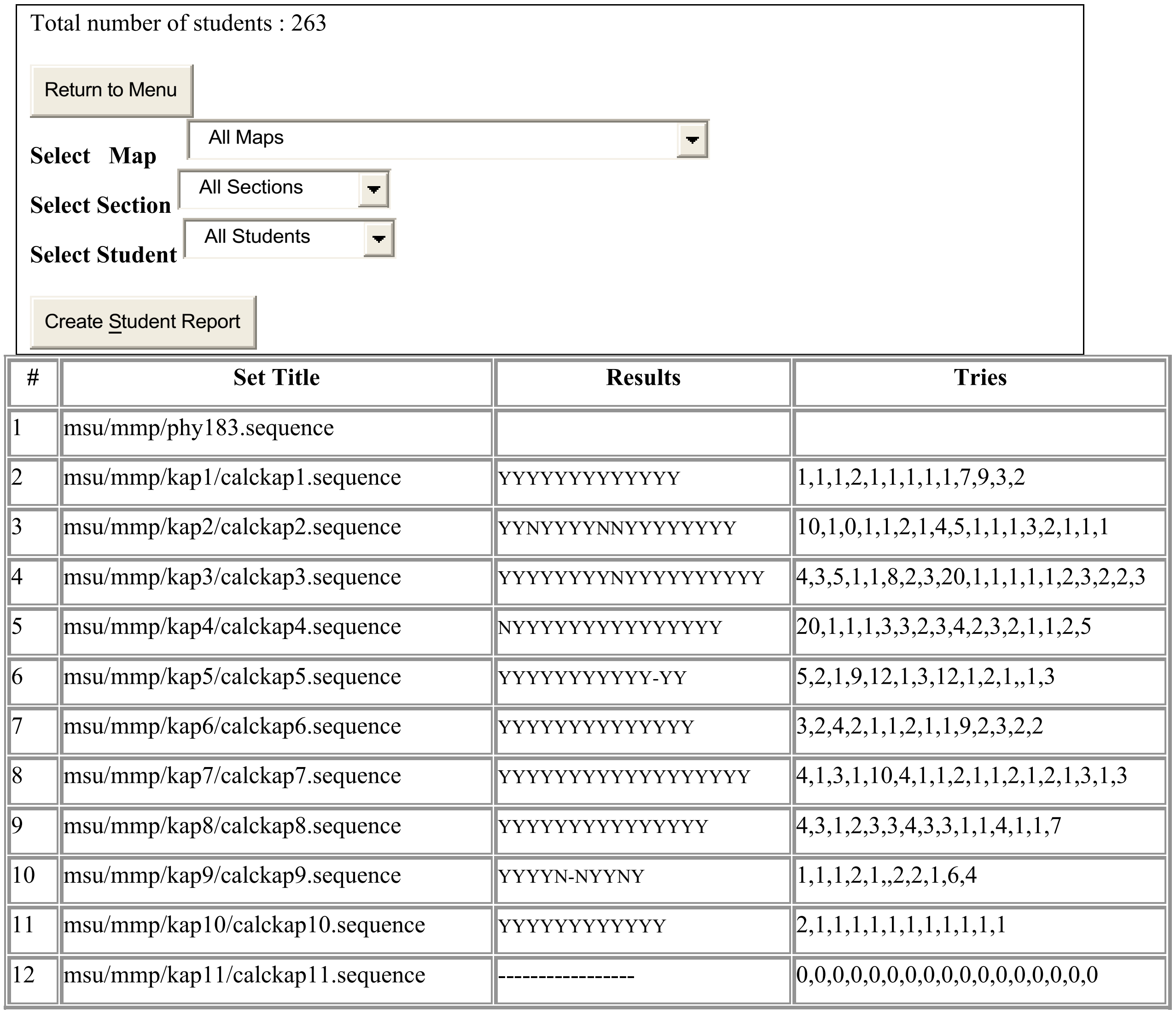
#	Concept	Correct	Wrong
1	Two masses have same acceleration if the two the string does not stretch.	619	147
2	Weight of the two masses is greater than the tension of the string attached to the ceiling.	256	510
3	The top tension is equals the two bottom tensions. (massless pulley)	529	237
4	Tension holding the two masses are equal if mass of pulley=0	465	301

5	Sub-System accelerates upwards or downwards accordingly	338	428
6	Center of mass accelerates downward	609	157
	From:[Fri Feb 1 00:15:26 2002] To: [Mon Feb 4 23:59:59 2002]	2816	1775

Fig. 3.2. 10: Table of student tries for Atwood Machine Problem according to every concept in 3 times interval.

Student Assessment

This option provides some reports about the current educational situation of every student as you see in **Fig. 3.2.11**. A 'Y' show that the student has solved the problem and 'N' shows his/her failure. A '-' denotes a unattempted problem. The numbers in the right column show the total number of tries of the student in solving the corresponding problems.


Total number of students : 263

#	Set Title	Results	Tries
1	msu/mmp/phy183.sequence		
2	msu/mmp/kap1/calckap1.sequence	YYYYYYYYYYYYY	1,1,1,2,1,1,1,1,1,7,9,3,2
3	msu/mmp/kap2/calckap2.sequence	YYNYYYYNNYYYYYYYY	10,1,0,1,1,2,1,4,5,1,1,1,3,2,1,1,1
4	msu/mmp/kap3/calckap3.sequence	YYYYYYYYNYYYYYYYYYY	4,3,5,1,1,8,2,3,20,1,1,1,1,1,2,3,2,2,3
5	msu/mmp/kap4/calckap4.sequence	NYYYYYYYYYYYYYYY	20,1,1,1,3,3,2,3,4,2,3,2,1,1,2,5
6	msu/mmp/kap5/calckap5.sequence	YYYYYYYYYYY-YY	5,2,1,9,12,1,3,12,1,2,1,,1,3
7	msu/mmp/kap6/calckap6.sequence	YYYYYYYYYYYYYY	3,2,4,2,1,1,2,1,1,9,2,3,2,2
8	msu/mmp/kap7/calckap7.sequence	YYYYYYYYYYYYYYYYYY	4,1,3,1,10,4,1,1,2,1,1,2,1,2,1,3,1,3
9	msu/mmp/kap8/calckap8.sequence	YYYYYYYYYYYYYYY	4,3,1,2,3,3,4,3,3,1,1,4,1,1,7
10	msu/mmp/kap9/calckap9.sequence	YYYYN-NYYNY	1,1,1,2,1,,2,2,1,6,4
11	msu/mmp/kap10/calckap10.sequence	YYYYYYYYYYYY	2,1,1,1,1,1,1,1,1,1,1,1
12	msu/mmp/kap11/calckap11.sequence	----------------	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Fig. 3.2. 5: A sample of a student homework results and tries

We plan to present several other reports in this page as well. Some student classification reports also would be depicted here.

In "problem stats" and "student assessment" pages, an instructor can limit the range of his/her information to a particular section or a particular homework set. So, an instructor would be able to load the statistics table or student tries in "student assessment" table to a particular map or section. After changing the map or section, he/she would be able to see the results via the recalculating the computation on that page.

Future work, using activity.log to classify the students

The problem is whether we can find the good features for *classifying* students? If so, we would be able to identify a *predictor* for any individual student after doing a couple of homework sets. With this information, we would be able to *help* a student use the resources better. As the first step of data mining stuff we want to make an initial effort to classify the students.

Preprocessing and finding the useful student data and segmenting may be a difficult task. Internally, one part of this data is stored in a student directory:
`/home/httpd/lonUsers/`*`domain/1st.char/2nd.char/3rd.char/username/`*
For example: /home/httpd/lonUsers/msu/m/i/n/minaeibi/
Since spring semester 2002, LON-CAPA has logged every activity of every student who has used online educational resources and their recorded paths through the web. So another part of data is stored in "activity.log" which is located in course directory.

The student data restored from .db files in student directory and is fetched into a hash table. The special hash keys "*keys*", "*version*" and "*timestamp*" were evaluated from the hash. The *version* will be equal to the total number of versions of the data that have been stored. The *timestamp* attribute is the UNIX time the data was stored. *keys* is available in every historical section to list which keys were added or changed at a specific historical revision of a hash. We extract the features from a structured homework data, which is stored as particular URL's. For example the result of solving homework's problem by students could be extracted from *`resource.partid.solved`*, the total number of the students for solving the problem could be extracted from *`resource.partid.tries`*, and so forth.

All data stored in activity.log includes *user name*, *time* and *resource URL*. We can divide these data into six types of URLs, listed below, according to their importance for data mining:

1. **problems:** are the most useful data. i.e.
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
2. **html pages** to those are some links in the problems. i.e.
msu/mmp/kap14/kap14.sequence___5___msu/mmp/kap14/cd396.htm
3. *the images*, which are loaded in above html pages. i.e. */res/msu/mmp/kap14/picts/backsoun.gif* .

4. loncapa routines: i.e. /adm/navmaps, or /adm/roles, or /adm/logout.

5. *Posted data* by students: i.e. *resource.0.11.submission=27.11*

6. remote control gif files: i.e.: /res/adm/pages/v.gif

So, activity.log usually grows fast, when students have more access to the educational resources. We have brought a sample of different types of data which are logged in activity.log after a preprocessing phase as follows:

```
144) 1010955846: studentX --> /adm/navmaps
145) 1010955205: studentX --> /res/msu/mmp/kap14/picts/beta_eqn.gif
146) 1010955685: studentX --> /adm/navmaps
147) 1010955988: studentX --> /adm/navmaps
148) 1010955998: studentX --> msu/mmp/kap14/kap14.sequence___5___msu/mmp/kap14/cd396.htm
149) 1010955999: studentX --> /res/msu/mmp/kap14/picts/velocity_eqn3.gif
150) 1010956000: studentX --> /res/msu/mmp/kap14/picts/time_eqn.gif
151) 1010954609: studentX --> /res/adm/pages/grds.gif
152) 1010954611: studentX --> /res/msu/mmp/wordproc.gif
153) 1010954626: studentX --> /res/adm/pages/i.gif
154) 1010955717: studentX --> msu/mmp/kap14/kap14.sequence___1___msu/mmp/kap14/cd392.htm
155) 1010955717: studentX --> /res/msu/mmp/kap14/picts/backsoun.gif
156) 1010955920: studentX --> msu/mmp/kap14/kap14.sequence___3___msu/mmp/kap14/cd394.htm
157) 1010955921: studentX --> /res/msu/mmp/gifs/demo.gif
158) 1010956113: studentX --> /res/adm/pages/v.gif
159) 1010954629: studentX --> /res/adm/pages/eval.gif
160) 1010954631: studentX --> /res/adm/pages/back.gif
161) 1010954632: studentX --> /res/adm/pages/b.gif
162) 1010954633: studentX --> /res/adm/pages/r.gif
163) 1010955754: studentX --> msu/mmp/kap14/kap14.sequence___2___msu/mmp/kap14/cd393.htm
164) 1010955756: studentX --> /res/msu/mmp/kap14/picts/asound.jpg
165) 1010955762: studentX --> /res/msu/mmp/kap14/picts/sensor.jpg
166) 1010955999: studentX --> /res/msu/mmp/gifs2/example.gif
173) 1010955687: studentX --> /res/adm/pages/u.gif
174) 1010955688: studentX --> /res/adm/pages/s.gif
175) 1010955688: studentX --> /res/adm/pages/e.gif
176) 1010956528: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
177) Sent data
178) 1010956536: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
179) Sent data HWVAL11=27.11
180) 1010956536: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
181) Sent data resource.0.11.submission=27.11
182) 1010956702: studentX -->
msu/mmp/kap14/kap14.sequence___33___msu/mmp/kap14/problems/cd418a.problem
183) Sent data
184) 1010955921: studentX --> /res/msu/mmp/kap14/picts/areal.gif
185) 1010955731: studentX --> /res/adm/pages/n.gif
```

- **Feature Extraction**

From these two types of student data which are stored by the LON-CAPA system, the features may be considered for classifying the students, are as follows:

1. Total number of correct answers.
2. Total number of tries for doing homework.
3. Time at which the student got the problem correct. Usually better students get the homework completed earlier.
4. Reading the material before attempting homework vs. attempting first and then read up on it.
5. Submitting a lot of attempts in a short amount of time without looking up material in between, vs. those giving it one try, reading up, submitting another one, etc.
6. Getting the problem right on the first try, vs. those with high number of tries.
7. Giving up on a problem versus students continuing trying up to the deadline.
8. Participating in the communication mechanisms, vs. those working alone.

It might be interesting to group students with time of the first log on (beginning of assignment, middle of the week, last minute) and correlate this with the number of tries or number of solved problems. A student who gets all correct answers will not necessarily be in the successful group if they took an average of 5 tries per problem, but it should be verified from this research.

We hope to find similar patterns of use in the data gathered from LON-CAPA, and eventually be able to make predictions as to the most-beneficial course of studies for each learner based on a limited number of variables for each individual student. Based on the current state of the learner in a learning sequence, the system could then make suggestions to the learner as to how to proceed.

Session Three: lonsql (Gerd)

This section describes issues associated with LON-CAPA and a SQL database.

The SQL database in LON-CAPA is used for catalog searches against resource metadata only. The authoritative version of the resource metadata is – as discussed – an XML-file on the normal file system (same file name as resource plus ".meta"). The SQL-database is a cache of these files, and can be reconstructed from the XML files at any time.

The current database is implemented assuming a non-adjustable architecture involving these data fields (specific to each version of a resource).

1. title
2. author
3. subject
4. notes
5. abstract
6. mime
7. language
8. creationdate
9. lastrevisiondate
10. owner
11. copyright

Purpose within LON-CAPA

LON-CAPA is meant to distribute A LOT of educational content to A LOT of people. It is ineffective to directly rely on contents within the ext2 filesystem to be speedily scanned for on-the-fly searches of content descriptions. (Simply put, it takes a cumbersome amount of time to open, read, analyze, and close thousands of files.)

The solution is to hash-index various data fields that are descriptive of the educational resources on a LON-CAPA server machine. Descriptive data fields are referred to as "metadata". The question then arises as to how this metadata is handled in terms of the rest of the LON-CAPA network without burdening client and daemon processes. I now answer this question in the format of Problem and Solution below.

PROBLEM SITUATION: If Server A wants data from Server B, Server A uses a lonc process to send a database command to a Server B lond process.

lonc= loncapa client process   A-lonc= a lonc process on Server A
lond= loncapa daemon process

```
                 database command
    A-lonc  --------TCP/IP----------------> B-lond
```

The problem emerges that A-lonc and B-lond are kept waiting for the MySQL server to "do its stuff", or in other words, perform the conceivably sophisticated, data-intensive, time-sucking database transaction. By tying up a lonc and lond process, this significantly cripples the capabilities of LON-CAPA servers.

While commercial databases have a variety of features that ATTEMPT to deal with this, freeware databases are still experimenting and exploring with different schemes with varying degrees of performance stability.

THE SOLUTION: A separate daemon process was created that B-lond works with to handle database requests. This daemon process is called "lonsql".

So,

```
                database command
  A-lonc  ---------TCP/IP-----------------> B-lond =====> B-lonsql
         <--------------------------------/                  |
           "ok, I'll get back to you..."                     |
                                                             |
                                                            /
  A-lond  <------------------------------  B-lonc   <======
           "Guess what? I have the result!"
```

Of course, depending on success or failure, the messages may vary, but the principle remains the same where a separate pool of children processes (lonsql's) handle the MySQL database manipulations.

Session Four: CVS/Bugzilla Intro (Guy)

Session Five: Open Worktime

Different Types of LON-CAPA Servers

Scott Harrison, freeware volunteer, sharrison@sourceforge.net

Introduction

There are two different kinds of LON-CAPA servers, Library Servers and Access Servers. On any LON-CAPA machine, you can look at the server type by viewing the file `/etc/httpd/conf/loncapa.conf` and looking at the variable **lonRole**.

Library Servers

Library Servers are repositories of authoritative educational resources. These servers also provide the construction space by which instructors assemble their classroom online material. First-time installations of LON-CAPA at an institution should be for Library Servers (and not Access Servers).

Only Library Servers make use of a MySQL database.

On the filesystem, the following directories are used primarily by Library Servers (and not Access Servers):

- `/home/USERNAME/` - authors are standard Linux users that exist on Library Servers
- `/home/httpd/lonUsers/` - user profile information is stored inside Library Servers
- `/home/httpd/html/res/`

  - Library Servers use this space to hold authoritative or cached versions of educational resources; Access Servers only use this space to hold cached versions of educational resources

Access Servers

Access Servers load-balance high-traffic delivery of educational resources over the world-wide web. Access Servers do not require a MySQL database and do not contain Linux user accounts. Several of the active directories on Library Servers are not needed on an Access Servers.

Hardware specification for LON-CAPA servers

Scott Harrison, freeware volunteer, sharrison@sourceforge.net

General information on LON-CAPA Server Hardware Requirements

For full-time, classroom usage, an adequate LON-CAPA server should be/have:

- a multiprocessor machine,
- a CPU speed of 1 gigahertz,
- a gigabyte of memory,
- and at least 40 to 80 gigabytes of hard disk space.

A lesser machine can be used for toying around with LON-CAPA (LON-CAPA will run for a single user on most any machine).

LON-CAPA servers experience significant peaks of activity before a homework submission deadline. To support these critical peaks of activity, it is strongly advised that LON-CAPA machines fit the above recommendation.

If thousands of students start accessing the box as a web server... well you may want to consider more options. The design of the LON-CAPA system is to naturally and transparently load-balance on multiple computer clusters. So, a simple solution for running an entire college campus is to just have an adequate plurality of LON-CAPA servers rather than a single, particularly monstrous server.

We like to think of high web server usage as "a good problem" though....

Additional Information on LON-CAPA Server Hardware Requirements

LON-CAPA works nicely (for development purposes) on a Pentium II, 20 gigabytes of hard-disk space, 256M RAM, and 400MHz. The consensus is though, that this may only be adequate for a class of a dozen students.

If you are making a serious investment, you should join the LON-CAPA mailing list by visiting http://mail.lon-capa.org/mailman/listinfo/lon-capa-users. By posting to this mailing list, you can learn about the solutions have worked for others.

LON-CAPA works on any Intel-based RedHat-compatible hardware. Unlike other e-learning software systems, it works comparatively well on dusty old crippled machines without much RAM or processing.

LON-CAPA on the Linux Filesystem

Scott Harrison, freeware volunteer, sharrison@sourceforge.net
June 4, 2002

Contents 1. Introduction 2. The LON-CAPA Source Files 3. Active Filesystem Directories of Importance 4. The Installation Mechanism 5. Automated Testing 6. Software Packages 7. Security on the Filesystem

1. Introduction

LON-CAPA currently consists of 695 files placed in 56 different directories on a linux filesystem. There are also 9 different symbolic links. There are 11 different file ownership/permission categories under which files are securely installed.

For the last 2 years, an average of 1 new file per day was included into the LON-CAPA distribution. A conservative estimate is that 10 files were modified each day in the LON-CAPA source tree. In order to maintain a robust build-installation framework consistent with the many changes, all information pertaining to the LON-CAPA installation has been kept in a well-formatted XML-valid document. The XML document "renders"

LON-CAPA from a CVS source tree to the Linux filesystem (the XML document can also be used to build RPMs, display documentation, and conduct status-checking reports). There have been few, if any, bugs related to the XML specification of the LON-CAPA software. Thus, throughout the software development and version releases of LON-CAPA, there has most always been a complete, bug-free specification of what files comprise the LON-CAPA software. For more information about this XML-based approach, please visit http://lpml.sourceforge.net/.

LON-CAPA is well-supported on RedHat Linux operating systems version 6.2 and version 7.3. LON-CAPA is also known to run well on Mandrake and Debian, however there is no specified installation procedure for these operating systems (a Debian installation procedure is currently under development). There currently do not exist LON-CAPA installations on BSD-UNIX or MacOS/X systems (I remain curious though as to whether a programmer or institution will pioneer this strategy). Given the current level of new LON-CAPA feature requests and frequency of new software versions, we suggest that most users plan on using RedHat so as to reduce high-frequency system administration overhead.

LON-CAPA is interdependent with a variety of Linux operating system services and software packages. Thus, a challenge exists to maintain the correct set of software packages (e.g. with RedHat, software packages are called RPMs) with the correct configurations. Past efforts have sought to completely automate the handling of software packages and their configuration. The new existing strategy is to test for proper operation of system services (architecturally termed as "horizontal layers") and make recommendations (to a knowledgeable linux administrator) as to how software packages or configurations should be adjusted.

2. The LON-CAPA Source Files

Of the 695 files which make up the LON-CAPA distribution, 492 of these files can be thought of as interface widgets or templates. The widgets and templates are organized into ten different file "globs".

1. Template files for generating new resources (28 files)
   source-to-target: loncapa/loncom/homework/templates/* -> /home/httpd/html/res/adm/includes/templates/

2. Icons for providing a HTML-tabled view of a course map (109 files)
   source-to-target: loncapa/rat/images/*.gif -> /home/httpd/html/adm/rat/

3. Icons to indicate an unexpected result (6 files)
   source-to-target: loncapa/loncom/html/adm/lonKaputt/*.* -> /home/httpd/html/adm/lonKaputt/

4. Logos and general widget icons (108 files)
   source-to-target: loncapa/loncom/html/adm/lonIcons/*.* -> /home/httpd/html/adm/lonIcons/

5. Miscellaneous resources (9 files)
   source-to-target: loncapa/loncom/html/adm/lonMisc/*.* -> /home/httpd/html/adm/lonMisc/

6. XML files which assign unicode numbers to mathematical symbols using <! ENTITY...> type syntax (26 files)
   source-to-target: loncapa/loncom/MathML/*.ent -> /home/httpd/html/adm/MathML/
7. Icons used for the entire LON-CAPA user interface (80 files)
   source-to-target: loncapa/loncom/html/res/adm/pages/*.gif -> /home/httpd/html/res/adm/pages/
8. Icons used for directory indexing (89 files)
   source-to-target: loncapa/loncom/html/res/adm/pages/indexericons/*.gif -> /home/httpd/html/res/adm/pages/indexericons/
9. Icons used for the bookmark portion of the LON-CAPA user interface (23 files)
   source-to-target: loncapa/loncom/html/res/adm/pages/bookmarkmenu/*.gif -> /home/httpd/html/res/adm/pages/bookmarkmenu/
10. Files associated with the scheme of displaying bookmarks (3 files)
   source-to-target: loncapa/loncom/html/res/adm/pages/bookmarkmenu/*.html -> /home/httpd/html/res/adm/pages/bookmarkmenu/

A second grouping of files are the 124 scripts, documentation files, shared object libraries and other extraneous files for LON-CAPA.

A third (and highly important) grouping of files are the 77 mod-perl Apache web handler modules. These modules coordinate how the system interacts with the web interface delivered by the LON-CAPA system. For example, the spreadsheet, search engine, graphing, and homework formatting features of LON-CAPA are all transacted by different Apache web handler modules. More information on the LON-CAPA Apache web handlers can be read about in the section titled "LON-CAPA and the 77 Web Handlers".

In addition to the total 695 files of LON-CAPA, 9 symbolic links are also created. 7 of these links exist inside the /etc/rc.d/ directories and are used to facilitate the launching of LON-CAPA network services (i.e. `lonc` and `lond`). Another link makes LON-CAPA resources available in their "raw" format (/home/httpd/html/raw is linked to /home/httpd/html/res). Another link makes /etc/mime.types equivalent with /etc/httpd/conf/mime.types to support backwards compatibility with various software packages for various RedHat releases.

3. Active Filesystem Directories of Importance

Of the 56 different directory locations where LON-CAPA files are stored, there are about a dozen directory locations of high significance for both operating and understanding the LON-CAPA software

- **/home/httpd/lonTabs/**
  tab-formatted reference files;
  e.g. `hosts.tab` (cross-networked servers) and `spare.tab` (load-balancing servers)

- **/home/httpd/perl/**
  perl scripts for managing the LON-CAPA system;
  e.g. `lond` (remote TCP command interpreter)

- **/home/httpd/perl/logs/**
  location where system performance and error logs are kept;
  e.g. `lonnet.log` (messages from the LON-CAPA web layer); `lonnet.log.1`,

`lonnet.log.2` (time-cycled past versions of `lonnet.log`); `lond.pid` (the parent process of all the `lond` children processes)

- **/home/httpd/html/lon-status/**
  location where online logs are kept that are viewable over the world-wide web;
  e.g. `index.html` (a summary file of all the error messages, warning messages, and connection states) and `londchld/*` (files that contain status reports from each `lond` process)

- **/home/httpd/lib/perl/**
  location of LON-CAPA non-web mod-perl handlers;
  e.g. `localauth.pm` (module for handling institution-customized user-password authentication)

- **/home/httpd/lib/perl/Apache/**
  location of LON-CAPA web mod-perl handlers;
  see the section titled "LON-CAPA and the 77 Web Handlers"; e.g. `lonnet.pm` (server-side socket connections) and `lonsearchcat.pm` (the LON-CAPA search catalogue)

- **/home/httpd/sockets/**
  lonc-related sockets (client portals for sending data to other machines)

- **/home/httpd/lonUsers/**
  Home directories of local users. There is a five-tier structure to the directories: [domain]/[firstletter]/[secondletter]/[thirdletter]/[userid]. For example, a user "fredflint" at MSU would have his information located in lonUsers/msu/f/r/e/fredflint. Or, for example, a user "thelma" at Ohio University would have her information located in lonUsers/ohiou/t/h/e/thelma. Within this directory there are a variety of files which govern the scope of what a user can do within the LON-CAPA system. A passwd file determines the type of authentication mechanism to use. There also are files such as roles.hist (and its database compilation, roles.db) which specify the roles and privileges for a user in the LON-CAPA system.

- **/home/httpd/lonIDs/**
  cookie jar, the location where the server can validate browser cookies present in client-side browsers (such as Internet Explorer or Netscape Communicator)

- **/home/httpd/html/res/DOMAIN/**
  publication space (or cache) for a domain;
  e.g. Simon Frasier University resources are specified in the `/home/httpd/html/res/sfu/` directory (note that the 'sfu' abbreviation is defined in `/home/httpd/lonTabs/hosts.tab`)

- **/home/USER/**
  construction space for a user (standard linux user home directory);
  e.g. `/home/korte/`, `/home/lucasm/`, or `/home/jensen/`

4. The Installation Mechanism

4.1 Standard procedure

To install LON-CAPA on a running RedHat Linux operating system, the procedure is as described at http://install.lon-capa.org/. The procedure is intentionally limited to an initial one-line command (for simplicity's sake). To summarize:

Download the most current loncapa.tar.gz.

```
wget http://install.lon-capa.org/versions/current/loncapa.tar.gz
tar xzvf loncapa.tar.gz
```

```
cd loncapa
```

The **UPDATE** command will refresh your filesystem with all the latest LON-CAPA software.

```
./UPDATE
```

Upon running the `./UPDATE` command, the user is presented with questions regarding the set up of their LON-CAPA server. For instance, the following is a summary question which verifies the machine-specific configuration variables (that are stored inside `/etc/httpd/conf/loncapa.conf`):

```

=====================================================================================
This is now the current configuration of your machine. 1) Domain Name: msu 2)
Machine Name: rawhidel1 3) System Administrator's E-mail Address:
harris41@spock.lite.msu.edu 4) Role: library 5) Cache Expiration Time: 86400 6)
Server Load: 2.00 7) Everything is correct up above ENTER A CHOICE OF 1-6 TO
CHANGE, otherwise ENTER 7:
```

The formatting and valid ASCII character usage are interactively monitored by the UPDATE script.

4.2 XML specification

`loncapa/doc/loncapafiles/loncapafiles.lpml` is an XML file that contains all the information pertaining to the LON-CAPA installation. Even though `loncapa/doc/loncapafiles/loncapafiles.lpml` is present in the "doc" directory of the LON-CAPA source tree, it is the most critical file of the software build-installation process. `loncapa/doc/loncapafiles/loncapafiles.lpml` is referenced by many different `loncapa/loncom/build/Makefile` targets.

An example entry inside `loncapa/doc/loncapafiles/loncapafiles.lpml` is:

```
 <file> <source>rat/lonwrapper.pm</source> <target
dist='default'>home/httpd/lib/perl/Apache/lonwrapper.pm</target>
<categoryname>handler</categoryname> <description> Wrapper for external
and binary files as standalone resources. Edit handler for rat maps;
TeX content handler. </description> <status>works/unverified</status>
</file>
```

LPML is an XML document-type definition which is described at http://lpml.sourceforge.net/.

4.3 Makefile

`loncapa/loncom/build/Makefile` is the "engine" of the source code tree (i.e. the aforementioned `./UPDATE` script simply calls different `Makefile` targets consecutively). Ordinarily, only software developers bypass the `./UPDATE` script and utilize the `Makefile` targets directly.

There are currently 40 targets present in the `Makefile` (each target is invoked with the syntax "`make TARGETNAME`" from the `loncapa/loncom/build/` directory.. Of these targets, there are several targets of common usage. These common targets are described immediately below.

make **install**: this installs configuration files, non-configuration files, configures the Apache `httpd.conf` file, runs a sanity check on the operating system; generally speaking, this target does everything associated with installing and upgrading files on a LON-CAPA filesystem.

make **rawinstall**: this installs configuration files and non-configuration files (without extra bells-and-whistles such as configuring the `httpd.conf` file or running a sanity check).

make **configinstall**: this installs configuration files and is a sub-target of the rawinstall and install targets.

make **build**: this compiles all files that need to be compiled from the LON-CAPA source tree.

make **test**: this runs software functionality tests.

5. Automated Testing

The goal is to test for proper operation of system services and make recommendations (to a knowledgeable linux administrator) as to how software packages or configurations should be adjusted.

The importance of a diagnostic test tool is especially relevant when installing LON-CAPA on non-RedHat Linux systems, or for testing installation procedures for new RedHat version releases.

In terms of better developer-to-user interactions, the current experience is that automated testing has helped speed the identification of problems associated with the server configuration and software dependencies.

5.1 Standard Procedure for Testing

After running the `./UPDATE` command from the `loncapa.tar.gz` LON-CAPA distribution, the user should then run the `./TEST` command to ensure the continuing correct operation of the LON-CAPA software.

```
./TEST
```

Using the **TEST** command may be an iterative process. It is normal to expect that the `./TEST` command will recommend for users to perform various steps to ensure optimal performance of their LON-CAPA server.

5.2 Testing the System Services (MySQL, perl libraries, etc)

Two of the most important (yet occasionally tricky) system services to handle are those involving 1) the MySQL database and 2) the dozens of perl software packages needed from http://www.cpan.org/ (as described in `loncapa/doc/otherfiles/perl_modules.txt`).

The testing of these system dependencies is performed inside the `loncapa/loncom/build/system_dependencies/` directory.

MySQL testing checks for:

- the presence of a MySQL server
- the status of the MySQL server
- the necessary perl modules (such as `DBI`)
- and most importantly, the run-time ability to connect to the MySQL "loncapa" database through the `DBI` interface module.

In the event of failed testing, the user is prompted as to what changes to make to their operating system.

LON-CAPA relies on the presence of multiple CPAN (http://www.cpan.org/) perl modules which are not ordinarily distributed with RedHat operating systems. Examples include `Digest-MD5`, `Math-FFT`, `GDTextUtil`, and `Algorithm-Diff`.

In the event of missing perl modules, the user is prompted to update their system with either a LON-CAPA-systemperl-*.*-rh72.i386.rpm package or a LON-CAPA-systemperl-*.*-1.i386.rpm package.

5.3 Testing the LON-CAPA Web Interface

The testing of the LON-CAPA web layer is performed inside the `loncapa/loncom/build/weblayer_test/` directory.

Simulated web transactions are conducted; this ordinarily would be considered "black-box" testing. Currently, a test login has been implemented. The machine's host id and domain for LON-CAPA is determined (read from `/etc/httpd/conf/loncapa.conf`). A password is randomly generated and a test user with a name beginning with 'ZXQTEST' is created within the `/home/httpd/lonUsers/` directory. A test login is run by using the LWP::UserAgent perl module. Password encryption that ordinarily occurs by javascript is simulated with the perl Crypt::DES module.

In the event of a failed login, the system administrator is prompted with corrective suggestions such as "Are lonc and lond running on the system?".

5.4 Testing the Installation Mechanism

The testing of the installation is performed inside the `loncapa/loncom/test/` directory.

Since a great deal of the installation mechanism relies on the ability to compare the files inside the LON-CAPA source tree with the target filesystem, most of the tests relate to the file comparison utilities.

6. Software Packages

```
./CHECKRPMS
```

In addition to the `./TEST` and `./UPDATE` commands, there is a `./CHECKRPMS` command. The `./CHECKRPMS` command first attempts to contact different RedHat RPM mirrors. When successful contact is made, `./CHECKRPMS` compares the RPM version numbers (based on an intelligent method designed by Martin Siegert at Simon Frasier University) and outputs a list of RPMs that the system administrator should update.

Updating RPMs is an important task to do for security. Quite often, even those RPMs that are not directly associated with system networking can present security holes (e.g. imlib, zlib, diffutils, and man all have had security updates in the last year).

`./CHECKRPMS` relies upon Martin Siegert's `check-rpms` script which exists in the `loncapa/loncom/build/` source directory. For more information on the advanced usage of `check-rpms`:

```
cd loncapa/loncom/build; perldoc ./check-rpms
```

Some advanced usage hints:

To automatically update RPMs for a RedHat 6.2 system:

```
check-rpms -v -r --update -ftp
rufus.w3.org/linux/redhat/updates/6.2/en/os
```

To automatically update RPMs for a RedHat 6.2 system:

```
check-rpms -v -r --update -ftp
rufus.w3.org/linux/redhat/updates/6.2/en/os
```

To download needed RPMs for a RedHat 6.2 system:

```
check-rpms -v -r -d /var/tmp/rpms -dl -ftp
```

7. Security on the Filesystem

Multiple processes controlled by multiple entities exist on a LON-CAPA server system. On a typical LON-CAPA library server, many different members of an institution will have author accounts by which they can log onto the server. The goal is to prevent these different user accounts from accessing other user directories (or sensitive information files such as password-files) on a LON-CAPA system. Two things are done to accomplish this on the filesystem. 1) A typical user is a member of only his or her own group. 2) A user's home directory can only be read by the user or root or the web-server processes.

The web-server processes are owned by the 'www' user. In order for web-server processes to work with the home directories of each user, the 'www' user is made a member of each user's personal group. For example, on a system with korte, lucasm, and jensen as authors (and thus standard Linux account users with filesystem privileges), www would be a member of the 'korte' group, the 'lucasm' group, and the 'jensen' group as defined in the `/etc/group` and `/etc/group-` files.

Several system interactions require the 'www' user to have the ability of the all-powerful 'root' user. (Future plans are for this to be mediated by 'sudo'.) Currently, there are several setuid-root scripts described in `loncapa/doc/loncapafiles/loncapafiles.lpml`. These setuid scripts allow 'www' processes to add new system users, change passwords, and password authentication against shadow passwords.

In order to ensure proper access and execution rights to various LON-CAPA software files, multiple file ownership/permission categories are supported in `loncapa/doc/loncapafiles/loncapafiles.lpml` as shown in the table below.

Category Name	**Permissions (development)**
interface file	0644 www:www
setuid script	6755 root:root
handler	0600 www:www
static conf	0444 root:root
conf	0644 root:root
script	0700 www:www
graphic file	0400 www:www
doc	0644 root:root
system file	0644 root:root
root script	0700 root:root
symbolic link	root:root
standard	0755 root:root
server standard	0755 www:www
server readonly	0700 www:www

During a brand-new installation, in order to make LON-CAPA work on a system with shadow passwords, the system administrator must manually configure and compile a `mod_auth_external` software package. Instructions for this, as currently described on http://install.lon-capa.org/ are:

First ensure the existence of a 'www' user. If 'www' does not yet exist on the system, then create:

```
/usr/sbin/useradd www
```

Make a LON-CAPA system work with shadow passwords

Step #	Description
1	Is your system using shadow passwords? (Note: LON-CAPA will work with either MD5/non-MD5 configured systems). If your system is not using shadow passwords, then do not perform any of the additional steps. If your system is using shadow passwords, then you will need to perform the additional steps below. **How to detect:** command: `cat /etc/passwd | grep ':x:'` If there is output such as "`root:x:0:0:root:/root:/bin/bash`", then your system is using shadow passwords and you will need to continue with the steps below.
2	**Retrieve the mod_auth_external source** by running the following command `wget http://www.wwnet.net/~janc/software/mod_auth_external-2.1.13.tar.gz`
3	**Unpack the mod_auth_external source** by running the following command `tar xzvf mod_auth_external-2.1.13.tar.gz`
4	**Go to the `pwauth` directory** by running the following command `cd mod_auth_external-2.1.13/pwauth/`

5	**Edit `config.h` and change SERVER_UIDS definition**
	
	Determine the user id of 'www':
	`grep ^www /etc/passwd | cut -d':' -f3`
	Change the line
	`#define SERVER_UIDS 99 /* user "nobody" */`
	to be
	`#define SERVER_UIDS 513 /* user "www" */`
	where in this example 513 corresponds to the user id of 'www'.

6	**Compile the `pwauth` executable** by running the following command `make`

7	**Install `pwauth` by doing the following**
	
	`cp pwauth /usr/local/sbin/`
	`chmod 6755 /usr/local/sbin/pwauth`
	
	Edit (creating the file) /etc/pam.d/pwauth to have the contents:
	
	`auth       required     /lib/security/pam_pwdb.so shadow nullok`
	`auth       required     /lib/security/pam_nologin.so`
	`account    required     /lib/security/pam_pwdb.so`

Configuration

Scott Harrison, freeware volunteer, sharrison@sourceforge.net

Configuring the Apache Web Server

The configuring of the Apache Web Server and the input of LON-CAPA configuration values is handled automatically by the `./UPDATE` command. In this document, the details of what the `./UPDATE` command facilitates are described.

`/etc/httpd/conf/httpd.conf` is adjusted to include the LON-CAPA web server configuration. This happens by appending the following line to `httpd.conf`:

```
Include conf/loncapa_apache.conf
```

The `/etc/httpd/conf/loncapa_apache.conf` file customizes the Apache Web Server to have the needed configuration to deliver LON-CAPA on the world-wide web. The `loncapa_apache.conf` file does not contain any information specified by the system administrator. The `loncapa_apache.conf` file does, however, cause `loncapa.conf` to be included. `/etc/httpd/conf/loncapa.conf` contains only information specified by the system administrator. Generally speaking, the information inside `/etc/httpd/conf/loncapa.conf` is machine-specific (each LON-CAPA server has its own specific values inside `loncapa.conf`).

`/etc/httpd/conf/loncapa_apache.conf` performs the following customizations of the Apache Web Server:

- The document root of all the web pages is set to be `/home/httpd/html/`.
- The web httpd user is set to be 'www'. The web httpd group is set to be 'www'. This means that all web processes only have the authorities granted by the Linux system to user=www and group=www.
- Shared object modules are supported for the perl language (mod-perl).
- A print spool alias and a cgi-bin script alias are specified.
- Access handlers are specified for published educational resources, author construction spaces, and raw binary data access.
- 50 different URI syntaxes are assigned to web handlers present inside `/home/httpd/lib/perl/Apache/`.

- Directory options are set for '`/`' and '`/home/httpd/html/`'.
- General configuration values are specified that can be accessed by the perl web handler modules; e.g. lonTabDir=/home/httpd/lonTabs, londPort=5663, etc.
- `/etc/httpd/conf/startup.pl` is invoked to test and load the perl web handler modules into memory for the httpd web server processes.

`/etc/httpd/conf/loncapa.conf` specifies machine-specific values that are accessible to the perl web handler modules. Ordinarily, these values are not to be entered manually. These values are interactively solicited from the system administrator by the `loncapa/doc/loncapafiles/updatequery.piml` file. The following values comprise the `loncapa.conf` file:

- **LON Host ID ("Machine Name")**
  is an internal ID within the LON-CAPA network used to specify the uniqueness of a particular LON-CAPA server.
- **LON System Administrator E-Mail**
  specifies the e-mail address of an institutional member responsible for direct upkeep of this server.
- **LON Domain ("Domain Name")**
  is a unique internal identifier within the LON-CAPA network specific to the home institution; A LON-CAPA domain is a collection of load-balancing institutional machines forming a cluster of LON-CAPA servers. It has nothing to do with the many other meanings for the word "domain", such as "domain names" like "loncapa.org", or Windows file server domains. If you are installing a LON-CAPA server for testing purposes, it is safe to create your own domain name for testing. You should ask the person running a real domain before adding your computer to their domain.
- **LON Load Limit ("Server Load")**
  specifies a threshold of activity within The LearningOnline Network that this machine should provide. The recommended value is 2.00. Depending on processor architecture (dual processor), this value may be increased, but there is no readily available measure in this regard.
- **Cache Expiration Time**
  indicates, in seconds, how long distributed resources should be held in the server's cache when not being accessed by students, instructors, or any other class of user. The recommended value is 86400.

Configuring the LON-CAPA Network

In order for LON-CAPA to connect to other servers that provide educational resources or to balance the load of high-volume web-requests, there needs to be a way to access information that describes the other LON-CAPA servers present on the internet.

`/home/httpd/lonTabs/hosts.tab` defines all of the other machines that a given LON-CAPA server can interact with. (In a sense, this is a mechanism of security so that connections cannot be attempted from spurious entities.)

The format of `hosts.tab` is to define each LON-CAPA server on separate lines. Each line is field-separated by colons ':'.

An example entry in `hosts.tab` is:

```
vulon1:msuvu:library:loncapa3.vu.msu.edu:35.9.80.250
```

The first field is the LON Host ID (see the above description of `loncapa.conf`). The second field is the Domain Name. The third field is the server type (see the section "Different Types of LON-CAPA Servers"). The fourth field is a DNS-registered full IP name. The fifth field is the numeric IP address.

`/home/httpd/lonTabs/spare.tab` is a list of LON Host IDs corresponding to Access Servers (or Library Servers). `spare.tab` should not be the same on every LON-CAPA server machine. The purpose of this list is to specify other servers toward which user web requests can be re-directed. Current experience is that the servers on `spare.tab` are particularly active (and much needed) shortly before homework assignments are due....

An example of `spare.tab` relates to the Library Server **vulon1** which load-balances the requests onto two other Access Servers as shown in the example `spare.tab` below:

```
vulona1
vulona2
```

Configuring MySQL

The following commands describe how to configure the MySQL database on your LON-CAPA server.
Note:

- you should substitute 'ROOTPASSWORD' with something very hard to guess (it does not have to be the Linux OS root password)
- The MySQL www@localhost user must always have a password of 'localhostkey' in order for there to be correct operation of a standard LON-CAPA system.

The following instructions assume you are logged in as 'root'.

Entering the mysql shell

```
 mysql -u root -p mysql OR mysql -u root mysql (depending on whether
you have set a root password)
```

Creating the mysql 'www' user (after entering mysql shell)

```
 mysql> CREATE DATABASE loncapa;  mysql> INSERT INTO user (Host, User,
Password) mysql> VALUES ('localhost','www',password('localhostkey'));
mysql> GRANT ALL PRIVILEGES ON *.* TO www@localhost;  mysql> FLUSH
PRIVILEGES;
```

SECURITY: set a password for the mysql 'root' user

```
 shell> mysql -u root mysql mysql> SET PASSWORD FOR
root@localhost=PASSWORD('ROOTPASSWORD');
```

SECURITY: only allow access from localhost

```
 shell> mysql -u root -p mysql mysql> DELETE FROM user WHERE
host<>'localhost';
```

Fast Installation: Setting up a Red Hat 7.3 LON-CAPA Server

Before you begin

Installing Linux is getting easier and easier. However, it is still a non-trivial undertaking and experience with Red Hat Linux will make this process easier. You will be required to log in to the machine and execute some routine Unix commands. Some familiarity with Linux is assumed.

Installation Overview

The installation process takes the following steps:

1. Obtain Red Hat 7.3
2. Gather information for installing Red Hat
3. Install Red Hat 7.3
4. Determine your LON-CAPA Settings
5. Install LON-CAPA
6. Configure LON-CAPA
7. Pick a hosts.tab file
8. Create a Domain Coordinator
9. Start/Restart services
10. Log in to LON-CAPA

Obtain the Red Hat 7.3 installation CDs

If you like, you may contact the LON-CAPA development staff and we can mail the CDs to you. Another option is to download the cd images and burn them yourself, or install from a mirror site using ftp or http.

Determine Network settings for your site

You will need to have the following information for your site. You must have a static IP address (do not use DHCP).

- ip address
- netmask
- network
- broadcast
- hostname
- gateway
- domain name server(s)

Install Red Hat 7.3

When you install RedHat you will need to ensure the following (the list below is in the order the issues appear in the installation of Red Hat 7.3):

Installation Type
: You should definitely do a "Server" install. We do not recommend Trying to install LON-CAPA with a different installation type.

Partitioning your Drive

You may want to use the automatic partitioning feature of Red Hat, however you should review the results and be prepared to modify them. LON-CAPA resource files are stored in the /home directory, so the lion's share of the drive should be allocated here. If you have 8 GB of space for Red Hat, /home should receive 4 to 6 gigs. Be sure to include adequate swap space. A minimum is 512 Megs, but you should typically have 1 or 2x as much swap space as you do physical RAM.

Network Configuration

LON-CAPA will **not** work with a machine set up to use a dynamic IP address. When configuring your network card, be sure to unselect the DHCP option and enter in your network information.

Firewall Configuration

The installation script will remove and disable your firewall.

Package Group Selection

At a minimum, install the following packages:

- NFS File Server
- Windows File Server
- Web Server

Other packages can be installed as you like.

X Windows Configuration

If you chose to install the X-windows packages you will need to configure them. If you are unsure of the support for your video card you may wish to skip the configuration.

Other notes:

- Use md5 and shadow passwords if you are given the choice (this is the default).

Determine LON-CAPA settings

LON-CAPA requires a number of identifying parameters be set in order for it to function at all. Below is a list with descriptions.

Host Type (library or access)

The server must be designated a 'library' or an 'access' server. In general you should have a library server for your instructors to create their course content on and run their courses. Students should connect to access servers. If you are doing the first install of LON-CAPA at your site, or if you are playing with it for your own edification, you should make your machine a 'library' server.

LON-CAPA domain

Each site or school which installs LON-CAPA needs its own domain. Here at MSU we use 'msu'. You should choose something short but meaningful. *Restriction: One word, no hyphens, underscores, or special characters!*

LON-CAPA host id

Each LON-CAPA server requires a unique internal name. We use names such as "msul1" for the first library server. *Restriction: One word, no hyphens, underscores, or special characters!*

Host administrator email

The amount of email sent to this address is relatively minimal. Messages are sent every time the system starts up, or if the system is in serious trouble. On a laptop, make this `root@localhost`.

Root password for SQL database

In order to keep people from corrupting the MYSQL database, a root password is needed. You'll need to remember this.

Install LON-CAPA

Log in as root with the password you provided during the Red Hat installation process.

```
 wget http://install.loncapa.org/versions/rh73install/loncapa-
rh73install.tar tar xf loncapa-rh73install.tar cd install ./install.pl
```

Configure LON-CAPA

Execute the following commands

```
 cd /root/loncapa/ ./UPDATE
```

You will need to enter the LON-CAPA configuration information you requested above. Additionally, you will need to enter the following information (options 5 and 6 in the menu, recommended settings):

Cache Expiration Time
    86400
Server Load
    2.0

You can deviate from the above settings if you know what you are doing.

Picking a hosts.tab file

The "hosts.tab" file controls which LON-CAPA servers your server attempts to access and which servers it will respond to. If this is your first install of LON-CAPA on your network, we suggest choosing a 'standalone' "hosts.tab". On a laptop, you must choose 'standalone'. Please contact the support staff at www.loncapa.org for information about adding more hosts to your "hosts.tab" file.

Note: for **non**-standalone configurations, you do need to add yourself to the `/home/httpd/lonTabs/hosts.tab` to initially test your configuration. Eventually, you will be added to the general cluster tables by LON-CAPA staff, so this is temporary. A `hosts.tab` entry has the format

```
 lonHostID:domain:hosttype:dns-name:ipaddress
```

for example

```
 10315:library:s17.lite.msu.edu:35.8.63.27
```

Please edit this file with a UNIX editor, not a PC or Mac one, so that the linebreaks are correct.

Creating a Domain Coordinator

You will need at least one user at your site who has the role of 'domain coordinator'. This user creates accounts for other users and grants them additional privileges. The make_domain_coordinator.pl script invoked below requires that you enter the users password. The password will show in plaintext as you type it. Feel free to use the "passwd username" command to change it later. Replace USERNAME and DOMAIN with an appropriate user name and your domain.

```
 cd /root/loncapa/loncom/build perl make_domain_coordinator.pl USERNAME
DOMAIN (WILL PROMPT FOR PASSWORD HERE) mkdir ~USERNAME/public_html
chown USERNAME:www ~USERNAME/public_html chmod 0775
~USERNAME/public_html
```

Start/Restart Services

The services take about 10 minutes to start.

```
 /etc/init.d/loncontrol start /etc/init.d/httpd restart
```

Make Sure that Services Autostart on Boot

On Redhat 7.3 it is not guaranteed that Apache and MySQL automatically restart on boot, even when installing with "server" configuration. After logging in as root, check under "Programs" - "System" - "Service Configuration" that "Start on Boot" is checked for httpd and mysqld, save and restart..

Log in to your LON-CAPA Machine

Point a web browser at your new machine and log in as the domain coordinator. Congratulations!

Glossary

Server – The Network has two classes of servers, library servers and access servers

Home Server of a user – Server that stores all personal records and resources of a user

Library Server – A library server can act as a Home Server that stores all personal records of user, and is responsible for the initial authentication of that user when a session is opened on any server in the Network. For Authors, it also hosts their construction area and the authoritative copy of every resource that was published by that author. Library servers can be used as backups to host sessions when all access servers in the Network are overloaded.

Access Server – machines that host learner sessions.

Domain – The Network is divided into domains – domains could be defined by departmental or institutional boundaries. Domains provide the possibility to limit the spread of personal user information flow, load balancing, and content material. Physically, every domain needs at least one dedicated library server.

Users – users are identified by a usernames and the domain. Usernames need to be unique within a domain and are not coupled to specific courses. They can also be carried over several semesters.

Role - Every user can have several roles, and the roles can change over the lifetime of a username. For example, over the course of studies, a student username assumes the role of "student" in different courses.

Resource – multimedia piece of content

Rendering – the process of generating output from resources

Granularity – grain size of a resource. The system distinguishes between four generic levels of granularity: fragment, page, sequence and course

Fragment – a resource of the smallest self-contained renderable grain size. Examples: one image, one movie clip, one paragraph of text

Page – a resource of the grain size which would be rendered as one page on the web and/or on the printer

Sequence – a resource which would be rendered as a sequence of pages, not necessarily linear. Examples: one lesson, one chapter, one learning cycle

Course – a sequence which represents the entirety of the resources belonging to a learning unit into which learners can be enrolled. Examples: a University one-semester course, a workshop, a High School class. Courses are defined by a "course-level" resource assembly map and an enrollment list with sections.

Resource Priority – a resource can be “regular,” “mandatory” or “optional.” These resource priorities are only used in book-keeping of earned points by the learners. There are two additional resources priorities, “start” and “finish” which cannot be set by authors

Map – a page, sequence or course resource

Start Resource – the first resource or entry point of a map

Finish Resource – the last resource or exit point of a map

Link – path from one resource to another within a map

Condition – condition under which a link can be taken. Examples for variables used in conditions: course parameters, learner performance, learner preferences, date

Condition Priority – a condition can be used to recommend a certain link over others branching off from the same resource, to block a link if it is false, or to force a link over all others if true

Resource Assembly – the process of generating a map by referencing and linking resources. The system allows for assembly of fragments and pages into a page; fragments, pages, sequences into a sequence; fragments, pages, sequences into a course.

Resource Pool – the entirety of the resources stored and cataloged within the system itself

External Resource – a resource which is not part of the resource pool

Author – the author of a resource

Learner – the consumer of resource renderings

State – the entirety of the conditions which determine a particular learner's rendering of a course.

Linear Resource Assembly – the process of combining a set of resources into a non-branching non-variable order, where the rendering is independent of state. Examples: an image and a paragraph of text get assembled into one page where the image will always be rendered on top of the paragraph.