FileXfer
File Transfer Jobs

Raymond E. Marcil
<rmarcil@gci.com>

Revision 0.0.1 (July 1, 2016)

Abstract

The FileXfer application is a system for automated file transfer jobs for copying files. "There are 3 applications that make up the usage collection framework: `filexfer`, which does the actual file transfers; `filexfer-jobmonitor`, which is configured to monitor various aspects of jobs and create NMS alarms when necessary; and `filexfer-dataloader`, which bulk-loads file data into database tables. There are also housekeeping scripts called `filexfer-filearchive`, which keeps files in the data directory pruned and compressed, and `filexfer-fileunarchive`, which allows files to be pulled out of the archive so filexfer jobs can work with them again."[1]

[1] Usage Collection Framework (filexfer)

Contents

List of Figures

List of Tables

List of Definitions and Abbreviations

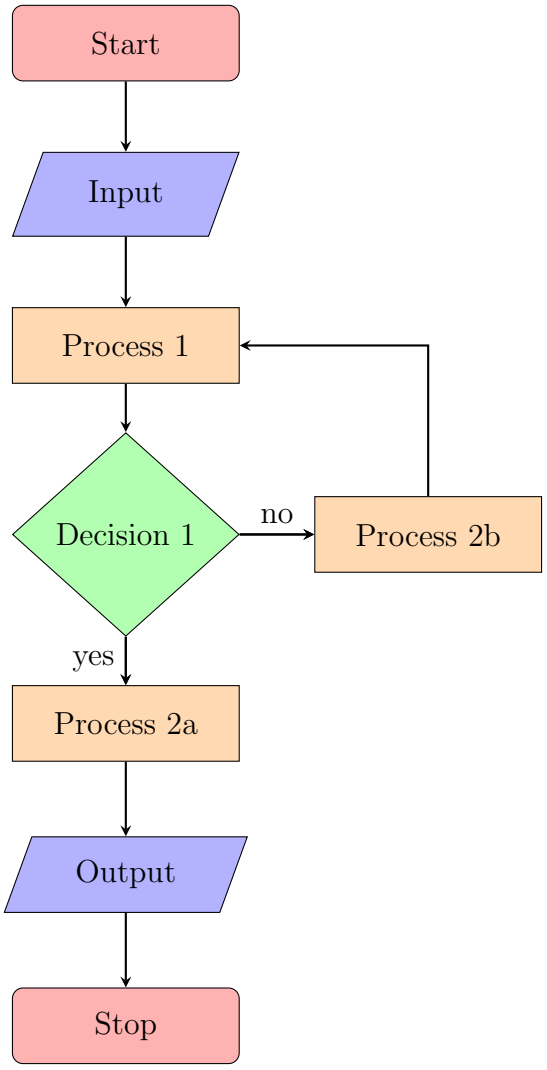
- **MOA** - Municipality of Anchorage

1 Introduction

The FileXfer system...

2 Design

[FIXME: Need data here...]



[FIXME: Customize flowchart FileXfer]

3 Schema

4 Implementation

[FIXME: Need data here...]

4.1 Configuration

[FIXME: Need data here...]

4.2 File Transfer Jobs

[FIXME: Need data here...]

4.3 Job Monitor

[FIXME: Need data here...]

4.4 Data Loader

[FIXME: Need data here...]

4.5 Logging

Application Logging

The filexfer applications log to the `/var/log/filexfer` directory on `prod-prov4-cdr1.operations.gci.com`. The parent `filexfer` jobs log to `filexfer-get.log` and `filexfer-put.log`. The `jobmonitor` and `dataloader` applications log to `jobmonitor.log` and `dataloader.log`, The `filexfer` applications log to the `/var/log/filexfer` directory on `prod-prov4-cdr1.operations.gci.com`. The parent `filexfer` jobs log to `filexfer-get.log` and `filexfer-put.log`. The `jobmonitor` and `dataloader` applications log to `jobmonitor.log` and `dataloader.log`, respectively. Each file transfer job is executed as a child process and gets its own log file. The format is `filexfer-{neName}-{idJob}-{get,put}.log`.

By default, the jobs log at the warn level. Adjust the level to info to get a high-level view of the application's state. Adjust log verbosity by modifying the appropriate config file in `/etc/filexfer`. The changes will take effect after the next program execution.

Errors are also logged to a database table which can be browsed in the filexfer web interface under the '`Logs & Errors`' view. This view includes messages logged at `warn`, `error`, and `fatal` severity.[2]

[2] Usage Collection Framework (filexfer)

File Transfer Logging

Every file transfer is recorded in a database table. There are two reasons for this table: first, it tells `filexfer` hich files have already been transferred, and second, it provides an audit trail for SOX compliance. The table is `filexfer.logs` on `sadc-cdr-mysql1.operations.gci.com`. Use the `filexfer.joblogs` view to easily find logs by job name or network element ID.

File transfer logs may also be viewed in the '`Logs & Errors`' page of the web interface.[3]

[3] Usage Collection Framework (filexfer)

4.6 Test

[FIXME: Need data here...]

4.7 Issues

[FIXME: Need data here...]

5 Operation

[FIXME: Need data here...]

5.1 Job Scheduling

Jobs are scheduled using a web interface at `nms.operations.gci.com/relevance`. Navigate to the "FileXfer" application and click the "File Transfer Jobs" link. Job execution happens on `prod-prov4-cdr1.operations.gci.com`. A `cron` job executes every minute from `/etc/cron.d/filexfer` to kick off the various `filexfer` scripts.[4]

Job Timing

The parent `filexfer` script is responsible for spawning child processes for each job. Since a large number of jobs can be scheduled at any given interval, the parent process limits how many children can run concurrently. As long as the limit is reached and more jobs need to be spawned, the parent process must stay alive. Since this may take longer than 1 minute, it is possible for `filexfer` to miss certain scheduling intervals.

For example, if 500 jobs are scheduled to run at the top of every hour `(0 * * * *)` and the maximum child process limit is 50, there is a good chance `filexfer` will not execute any jobs scheduled to run at 1 minute past the hour `(1 * * * *)`. The best way to avoid this is to use `0`, `15`, `30`, or `45` in the minute field of the job schedule. These intervals are always executed.[5]

5.2 Dataloader

Dataloader jobs are configured using the web interface at `nms.operations.gci.com/relevance`. Navigate to the "`FileXfer`" application and click the "`Data Load Jobs`" link. These jobs are executed every minute as long as there are files in the load queue.[6]

[4] Usage Collection Framework (filexfer)
[5] Job Timing
[6] Usage Collection Framework (filexfer)

Appendix

Source

There are 3 primary FileXfer `perl` scripts on `prod-prov4-cdr1`:[7]

File name	Attributes	Description
`filexfer.plx`	181 lines	File transfer jobs
`filexfer-dataloader.plx`	132 lines	Data loader
`filexfer-jobmonitor.plx`	132 lines	Job Monitor

Table 1: FileXfer `perl` scripts on `prod-prov4-cdr1`

[7] `prod-prov4-cdr1.operations.gci.com` (`192.168.161.47`, NATed IP: `66.223.199.228`), data including CDRs and such under `/data/usage/` — Network Services, OSS.

filexfer.plx — File Transfer Jobs

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer ();
7  our $VERSION = $Modules::App::FileXfer::VERSION;
8
9  # Core modules
10 use Clone qw( clone );
11 use File::Basename ();
12 use File::Spec ();
13 use POSIX ();
14
15 $SIG{CHLD} = \&Modules::App::FileXfer::REAPER;
16
17 MAIN: {
18     # Process and merge command-line and config file options
19     my $getopt    = Modules::App::FileXfer::
       ↪ get_command_line_options();
20     my $fileconf = Modules::App::FileXfer::read_config_file(
       ↪ $getopt->get_configfile );
21     Modules::App::FileXfer::merge_options( $getopt, $fileconf );
22
23     # Make sure we're the only instance running
24     Modules::App::FileXfer::check_pid_file( $Modules::App::
       ↪ FileXfer::Options->{pidfile} );
25
26     # Get logger and evenge objects
27     Modules::App::FileXfer::create_evenge_obj();
28     my $logger = Modules::App::FileXfer::create_logger_obj(
29         $Modules::App::FileXfer::Options->{logger}, $Modules::App
           ↪ ::FileXfer::Program );
30
31     # Get the ready jobs
32     my $fx       = Modules::App::FileXfer::create_filexfer_obj(
       ↪ $Modules::App::FileXfer::Program );
33     my $jobs     = Modules::App::FileXfer::get_jobs( $fx );
34     my $loadjobs = Modules::App::FileXfer::get_jobs_with_load_jobs
       ↪ ( $fx );
35     undef $fx;
36
```


by Raymond E. Marcil <rmarcil@gci.com>

```
37      for my $job ( @{ $jobs } )
38      {
39          # Enforce the "max children" constraint
40          $logger->info( 'Max child processes reached. Waiting for
                ↪ one to complete before starting job.' )
41              if ( scalar keys %Modules::App::FileXfer::Children
42                   >= $Modules::App::FileXfer::Options->{maxchildren
                         ↪ } );
43
44          sleep 1 while ( scalar keys %Modules::App::FileXfer::
                ↪ Children
45                          >= $Modules::App::FileXfer::Options->{
                              ↪ maxchildren} );
46
47          # Fork a child process for this job
48          $logger->info( sprintf( 'Spawning child process for job "%
                ↪ s".', $job->jobName ));
49
50          my $pid = fork;
51          defined $pid or Modules::App::FileXfer::log_event(
52              5, sprintf( "Can't fork for job \"%s\": %s", $job->
                    ↪ jobName, $! ), 'logdie' );
53
54          if ( $pid == 0 ) # child
55          {
56              # Set random seed for this child
57              srand();
58
59              # Lower the OS scheduling priority based on job
                    ↪ priority
60              POSIX::nice( Modules::App::FileXfer::pri_to_nice( $job
                    ↪ ->priority ));
61
62              # Add the NE name to our command line string
63              $0 .= " @ARGV " . $job->neName;
64              my $jobtag = Modules::App::FileXfer::get_jobtag( $job
                    ↪ );
65
66              # Set the subresource for this job in the evenge
                    ↪ object
67              $Modules::App::FileXfer::Evenge->subresourceName( $job
                    ↪ ->jobName );
68
69              # Create a logger specific to this child process
```

```
70              my $logopt = clone( $Modules::App::FileXfer::Options
                    ↪ ->{logger} );
71              my ( undef, $logdir ) = File::Basename::fileparse(
                    ↪ $logopt->{file}{filename} );
72              $logopt->{file}{filename} = File::Spec->catfile(
                    ↪ $logdir, "$jobtag.log" );
73
74              $logger->delete();
75              my $logger = Modules::App::FileXfer::create_logger_obj
                    ↪ ( $logopt, $job->jobName );
76              Log::Log4perl::MDC->put( 'idJob', $job->idJob );
77
78              # Make sure another instance isn't still running
79              Modules::App::FileXfer::check_pid_file( $jobtag );
80
81              # Create a FileXfer object for database updates
82              my $fx = Modules::App::FileXfer::create_filexfer_obj(
                    ↪ $jobtag );
83
84              # Execute the job
85              Modules::App::FileXfer::run_job( $fx, $job, $loadjobs
                    ↪ );
86
87              $logger->info( 'Child exiting.' );
88              exit 0;
89          }
90          else # parent
91          {
92              $logger->debug( sprintf( 'Spawned child process %d for
                    ↪  job "%s".', $pid, $job->jobName ));
93              $Modules::App::FileXfer::Children{ $pid } = $job->
                    ↪ jobName;
94          }
95      }
96
97      $logger->info( 'Main application exiting.' );
98  }
99
100 # Safely exit
101 $SIG{CHLD} = 'IGNORE';
102
103 __END__
104
105 =head1 NAME
```

```
106
107 filexfer -- Move a file from point A to point B over an IP network
108
109 =head1 VERSION
110
111 0.51
112
113 =head1 SYNOPSIS
114
115 filexfer.plx -c configfile -t {get|put} [options]
116
117 =head1 ARGUMENTS
118
119 =over 4
120
121 =item -c, --configfile
122
123 Specify the configuration file to load. Must be in YAML format.
124
125 =item -t, --transfertype
126
127 One of "get" or "put". Get jobs download files and put jobs upload
    ↪ files.
128
129 =back
130
131 =head1 OPTIONS
132
133 =over 4
134
135 =item -d, --piddir
136
137 Directory where the pid file will be written. Defaults to /var/run
    ↪ /filexfer.
138
139 =item --db
140
141 Sets the database connection parameters. Valid keys are: server (
    ↪ default
142 localhost), port (default 3306), driver (default mysql), uid, pwd,
    ↪ database,
143 and table. Specify tags as key/value pairs, e.g.:
144
145     --db server=localhost --db database=filexfer
```


by Raymond E. Marcil <rmarcil@gci.com>

```
146
147 =item -e, --evengehost
148
149 Address of the Evenge web server. Used to send indicators and
      ↪ events to the NMS system.
150
151 =item --evengetimeout
152
153 Timeout in seconds for communicating with the Evenge web server.
      ↪ Defaults to 10.
154
155 =item -f, --cachefile
156
157 Template cache file location. Defaults to /var/lib/filexfer/
      ↪ filexfer.kch.
158
159 =item -h, --help
160
161 Output this documentation.
162
163 =item -m, --maxchildren
164
165 Maximum number of child processes to spawn. Defaults to 50.
166
167 =item -p, --pidfile
168
169 PID file name. This will be appended with a ".pid" suffix.
170
171 =item -r, --resource
172
173 Resource name of this application. Used in indicator and event
      ↪ messages sent to the NMS system.
174
175 =item --verbose, -v
176
177 Log to the screen at increasingly verbose levels. This option may
      ↪ be repeated
178 multiple times to increase the log level. For example, "-v" logs
      ↪ at info level,
179 "-vv" logs at debug level, and "-vvv" logs at trace level.
180
181 =back
```

filexfer-jobmonitor.plx — Job monitor

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer::JobMonitor ();
7  our $VERSION = $Modules::App::FileXfer::JobMonitor::VERSION;
8
9  MAIN: {
10     # Process and merge command-line and config file options
11     my $getopt    = Modules::App::FileXfer::JobMonitor::
          ↪ get_command_line_options();
12     my $fileconf = Modules::App::FileXfer::JobMonitor::
          ↪ read_config_file( $getopt->get_configfile );
13     Modules::App::FileXfer::JobMonitor::merge_options( $getopt,
          ↪ $fileconf );
14
15     # Make sure we're the only instance running
16     Modules::App::FileXfer::JobMonitor::check_pid_file(
17          $Modules::App::FileXfer::JobMonitor::Options->{pidfile} );
18
19     # Get logger and evenge objects
20     Modules::App::FileXfer::JobMonitor::create_evenge_obj();
21     my $log = Modules::App::FileXfer::JobMonitor::
          ↪ create_logger_obj(
22          $Modules::App::FileXfer::JobMonitor::Options->{logger},
23          $Modules::App::FileXfer::JobMonitor::Program
24     );
25
26     # Get a list of job monitors
27     my $jm = Modules::App::FileXfer::JobMonitor::
          ↪ create_jobmonitor_obj();
28     my $mons = Modules::App::FileXfer::JobMonitor::get_monitors(
          ↪ $jm );
29
30     for my $mon ( @{ $mons } )
31     {
32          Log::Log4perl::MDC->put( 'idJob', $mon->idJob );
33          $log->info( sprintf( 'Executing monitor "%s".', $mon->
               ↪ monitorName ));
34
```


by Raymond E. Marcil <rmarcil@gci.com>

```
35          # Set the subresource for this monitor in the evenge
               ↪ object
36          $Modules::App::FileXfer::JobMonitor::Evenge->
               ↪ subresourceName( $mon->monitorName );
37
38          # Execute the job monitor
39          eval { Modules::App::FileXfer::JobMonitor::run_monitor(
               ↪ $jm, $mon ) };
40          $@ and $log->error( "$@" );
41      }
42
43      $log->info( 'Main application exiting.' );
44  }
45
46  __END__
47
48  =head1 NAME
49
50  jobmonitor -- Monitor filexfer jobs for any condition and generate
       ↪  alerts
51
52  =head1 VERSION
53
54  0.51
55
56  =head1 SYNOPSIS
57
58  jobmonitor.plx -c configfile [options]
59
60  =head1 ARGUMENTS
61
62  =over 4
63
64  =item -c, --configfile
65
66  Specify the configuration file to load. Must be in YAML format.
67
68  =back
69
70  =head1 OPTIONS
71
72  =over 4
73
74  =item -a, --mailhost
```

```
75
76 Address of the mail server. Used to send email notifications.
     ↪ Defaults to localhost.
77
78 =item -d, --piddir
79
80 Directory where the pid file will be written. Defaults to /var/run
     ↪ /filexfer.
81
82 =item --db
83
84 Sets the database connection parameters. Valid keys are: server (
     ↪ default
85 localhost), port (default 3306), driver (default mysql), uid, pwd,
     ↪  database,
86 and table. Specify tags as key/value pairs, e.g.:
87
88     --db server=localhost --db database=filexfer
89
90 =item -e, --evengehost
91
92 Address of the Evenge web server. Used to send indicators and
     ↪ events to the NMS system.
93
94 =item -f, --cachefile
95
96 Template cache file location. Defaults to /var/lib/filexfer/
     ↪ jobmonitor.kch.
97
98 =item -h, --help
99
100 Output this documentation.
101
102 =item -i, --mailinterval
103
104 Minimum time, in seconds, before repeat emails may be sent for the
     ↪  same monitor. Defaults to 3600 (1 hour).
105
106 =item -l, --mailfrom
107
108 Sender's email address in email notifications. Defaults to [
     ↪ username]@[hostname].
109
110 =item -m, --mailstatfile
```

```
111
112 Mail status file location. Defaults to /var/lib/filexfer/
    ↪ jobmonitor-mailstat.kch.
113
114 =item -o, --mailport
115
116 Port on which the mail server is listening for SMTP traffic.
    ↪ Defaults to 25.
117
118 =item -p, --pidfile
119
120 PID file name. This will be appended with a ".pid" suffix.
121
122 =item -r, --resource
123
124 Resource name of this application. Used in indicator and event
    ↪ messages sent to the NMS system.
125
126 =item --verbose, -v
127
128 Log to the screen at increasingly verbose levels. This option may
    ↪ be repeated
129 multiple times to increase the log level. For example, "-v" logs
    ↪ at info level,
130 "-vv" logs at debug level, and "-vvv" logs at trace level.
131
132 =back
```

filexfer-dataloader.plx — Data Loader

```
1  #!/usr/bin/perl
2
3  use strict;
4  use warnings;
5
6  use Modules::App::FileXfer::DataLoader ();
7  our $VERSION = $Modules::App::FileXfer::DataLoader::VERSION;
8
9  MAIN: {
10     # Process and merge command-line and config file options
11     my $getopt   = Modules::App::FileXfer::DataLoader::
          ↪ get_command_line_options();
12     my $fileconf = Modules::App::FileXfer::DataLoader::
          ↪ read_config_file( $getopt->get_configfile );
13     Modules::App::FileXfer::DataLoader::merge_options( $getopt,
          ↪ $fileconf );
14
15     # Make sure we're the only instance running
16     Modules::App::FileXfer::DataLoader::check_pid_file(
17         $Modules::App::FileXfer::DataLoader::Options->{pidfile} );
18
19     # Get logger and evenge objects
20     Modules::App::FileXfer::DataLoader::create_evenge_obj();
21     my $logger = Modules::App::FileXfer::DataLoader::
          ↪ create_logger_obj(
22         $Modules::App::FileXfer::DataLoader::Options->{logger},
23         $Modules::App::FileXfer::DataLoader::Program
24     );
25
26     # Get the load jobs with pending files
27     my $dl   = Modules::App::FileXfer::DataLoader::
          ↪ create_dataloader_obj();
28     my $jobs = Modules::App::FileXfer::DataLoader::get_load_jobs(
          ↪ $dl );
29
30     for my $job ( @{ $jobs } )
31     {
32         next if 414 == $job->idJob;
33         Log::Log4perl::MDC->put( 'idJob', $job->idJob );
34         $logger->info( sprintf( 'Executing job "%s".', $job->
              ↪ jobName ));
35
```

```
36          eval {
37              # Get the list of pending load files
38              my $files = Modules::App::FileXfer::DataLoader::
                    ↪ list_load_files( $dl, $job );
39              next unless scalar @{ $files };
40
41              # Import the class for this job's files
42              Modules::App::FileXfer::DataLoader::import_file_class(
                    ↪  $files->[0]->{fileclass} );
43
44              # Bulk load the data from each file
45              for my $file ( @{ $files } )
46              {
47                  Modules::App::FileXfer::DataLoader::load_file_data
                        ↪ ( $dl, $job, $file );
48                  Modules::App::FileXfer::DataLoader::
                        ↪ dequeue_load_file( $dl, $job, $file );
49              }
50          };
51
52          $@ and $logger->error( "$@" );
53
54          # Close the external db handle
55          $dl->close_ext_dbh();
56      }
57
58      $logger->info( 'Main application exiting.' );
59      exit;
60  }
61
62  __END__
63
64  =head1 NAME
65
66  filexfer-dataloader -- Bulk load file data from filexfer into a
        ↪ database table.
67
68  =head1 VERSION
69
70  0.51
71
72  =head1 SYNOPSIS
73
74  filexfer-dataloader -c configfile [options]
```

```
75
76 =head1 ARGUMENTS
77
78 =over 4
79
80 =item -c, --configfile
81
82 Specify the configuration file to load. Must be in YAML format.
83
84 =back
85
86 =head1 OPTIONS
87
88 =over 4
89
90 =item -d, --piddir
91
92 Directory where the pid file will be written. Defaults to /var/run
   ↪ /filexfer.
93
94 =item --db
95
96 Sets the database connection parameters. Valid keys are: server (
   ↪ default
97 localhost), port (default 3306), driver (default mysql), uid, pwd,
   ↪  database,
98 and table. Specify tags as key/value pairs, e.g.:
99
100     --db server=localhost --db database=filexfer
101
102 =item -e, --evengehost
103
104 Address of the Evenge web server. Used to send indicators and
   ↪ events to the NMS system.
105
106 =item --evengetimeout
107
108 Timeout in seconds for communicating with the Evenge web server.
   ↪ Defaults to 10.
109
110 =item -f, --cachefile
111
112 Template cache file location. Defaults to /var/lib/filexfer/
   ↪ dataloader.kch.
```


by Raymond E. Marcil <rmarcil@gci.com>

```
113
114 =item -h, --help
115
116 Output this documentation.
117
118 =item -p, --pidfile
119
120 PID file name. This will be appended with a ".pid" suffix.
121
122 =item -r, --resource
123
124 Resource name of this application. Used in indicator and event
    ↪ messages sent to the NMS system.
125
126 =item --verbose, -v
127
128 Log to the screen at increasingly verbose levels. This option may
    ↪ be repeated
129 multiple times to increase the log level. For example, "-v" logs
    ↪ at info level,
130 "-vv" logs at debug level, and "-vvv" logs at trace level.
131
132 =back
```

[FIXME: Need data here]

Table 2 – FileXfer directories and files on `prod-prov4-cdr1`

Directory	File(s)
/etc/filexfer/	*.conf
/usr/bin/	filexfer-dataloader
	filexfer-dataloader.plx
	filexfer-dataloader.plx.mbak
	filexfer-epg-dataloader.plx
	filexfer-filearchive
	filexfer-filearchive.sh
	filexfer-fileunarchive
	filexfer-fileunarchive.sh
	filexfer-jobmonitor
	filexfer-jobmonitor.plx
	filexfer.plx
/usr/lib/filexfer/	*.gz, *.sh, *.plx
	ExtractCarrierTurboZoneUsage*
/usr/share/filexfer/	filexfer.changelog-*.xml

Continued on next page

Table 2 – continued from previous page

Directory	File(s)
	filexfer.changelog-master.xml
	liquibase.sh
	.gnupg/pubring.gpg
	.gnupg/random_seed
/var/cache/yum/build/packages/	filexfer-0.52-1.el5.centos.noarch.rpm
/var/lib/filexfer/	dataloader.kch
	dataloader_temp.kch
	filexfer-aaa01-13-get.kch
	...
	filexfer-wps01-706-get.kch
	filexfer.kch
	jobmonitor-mailstat.kch
	jobmonitor.kch
/var/log/filexfer	*.log
	ExtractCarrierTurboZoneUsage_ACS.log
	archive
	convert-wps-om-counters-report-part2.log
	convert-wps-om-counters-report.log
	dataloader.log
	dataloader_temp.log
	dataloadinsert.log
	datarecovery.log
	epg-dataloader.log
	ericsson-oss-rl-reports-preprocess.log
	ericsson-oss-sts-reports-preprocess.log
	filearchive.log
	filexfer-aaa01-13-get.log
	filexfer-aaa01-14-put.log
	...
	/var/log/filexfer/filexfer-wps01-706-get.log

LaTeX - Examples and Formatting

Comments

> COMMENTS Comment — *Sean Weems, Spring 2003*
> We should get the COMMENTS column searchable via the landrecords application before we do much anything else – shouldn't be too hard.

> *Errata: Plats spanning multiple sections*
> A few anomalies can be observed in the AKPLATS table. Specifically plats exist that span multiple sections. Since the table only has a single column, SCODE, that accepts a single section code, SGU (Status Graphics Unit) has handled this problem by entering multiple rows in the table, each with a different section that point to the same plat or file. Multiple section plats are indicated by setting the TCODE column to the value 37, and making an appropriate notation like *Section 24-25-26-27* in the REMARKS column.
> [FIXME: Perhaps the SCODE column should accept an array of sections?]

Links

A Guide to LaTeX
http://www.astro.rug.nl/ kuijken/latex.html

LaTeX - From Wikibooks,the open-content textbooks collection
http://en.wikibooks.org/wiki/LaTeX

LaTeX Notes
http://luke.breuer.com/time/item/LaTeX_Notes/180.aspx