A

Conversion from Berkeley Mail Format to Cyrus: Tools

The procedure for converting a set of users from traditional Unix (Berkeley format) mail to Cyrus was outlined in Chapter 9, *Cyrus System Administration*. In this appendix, source code for the tools used in such a conversion is provided.

bsd2cyrus

bsd2cyrus is a Perl script, introduced in Chapter 9, that maps a set of users' Berkeley-format mail folders into the Cyrus namespace. The output of the *bsd2cyrus* script is used as input to other scripts that are used in converting users from a Berkeley-style mail system to a Cyrus system. *bsd2cyrus* takes as input the filename of a text file that contains a list of usernames. Example A-1 shows the *bsd2cyrus* script.

Example A-1. bsd2cyrus

```
#!/usr/local/bin/perl
    eval 'exec /usr/local/bin/perl -S $0 ${1+"$@"}'
        if $running_under_some_shell;

require "find.pl";

$inputfile  = "$ARGV[0]";
if (! $inputfile) { die "Usage: $0 inputfile\n"; }
```

For each user, get the user's home directory from the *passwd* file and search for the pathnames of all files under the user's *~/mail* directory. The `find` subroutine pushes those files onto an array. Note that the files are all pushed onto a single array, not an array per user—this is because we're assuming that you're converting a large batch of users all at one time and don't need to do anything special on a per-user basis.

```
open (DATA, $inputfile) || die "can't open $inputfile";
while (<DATA>) {
    chop;
    ($name,$pw,$uid,$gid,$quota,$cmnt,$gcos,$home) = getpwnam $_;
    next if $home eq "";
    &find("$home/mail");
}

close DATA;
```

The next lines narrow the list of files under the *~/mail* subdirectory to folders that contain mail content only. If your users store mail in directories named something other than *~/mail*, you should modify the script to look at those directories:

```
foreach (@folders) {

    ($user,$folder) = split(/:/,$_,2);
```

Before creating a mailbox on the Cyrus server, we have to check and make sure that the BSD folder on the old system has RFC 822 content. In the *bsd2cyrus* script, we assume that if the folder is any of the following:

- Directory
- Executable file
- Binary file
- Archive
- Empty file

then its content is not RFC 822. If you wish to use more or less stringent criteria—we chose to keep this example simple to illustrate the concept—in practice, you could also modify *bsd2cyrus* to log the skipped files, then go back and examine them more closely at a later time. Speaking from our own experience converting 20,000 accounts from Berkeley format on a UW IMAP system to Cyrus IMAP, the heuristic used in this example will handle 99.9% of "problem" mail folders. The more common problem we encountered was users who stored their mail folders in a non-standard *place* on the system, such as in their home directory, in system "scratch" space, or in a hidden subdirectory.

```
    if (! rfc822($folder) ) { next; }

    @tokens = split(/\//, $folder);
    $mailbox = $tokens[$#tokens];

    ## Sanity checks - earlier tests should have caught these.

    next if ($mailbox =~ /\.gz$/);      # Skip gzipped files
    next if ($mailbox =~ /\.Z$/);       # Skip compressed files
    next if ($mailbox =~ /^\./);        # Skip hidden files
```

You may recall from Chapter 6, *Introduction to the Cyrus IMAP Server*, that Cyrus IMAP does not allow special characters, such as Unix shell metacharacters and non-ASCII characters, in mailbox names. The *rm_badchars* subroutine handles special characters by converting them into their ASCII representation in the new mailbox name, preceded by an underscore:

```
    ## Replace "bad" characters with an underscore followed by
    ## the ASCII representation of the "bad" character.

    $mailbox = rm_badchars($mailbox);
    print "$user:user.$user.$mailbox:$folder\n";
}

sub wanted {
    (($dev,$ino,$mode,$nlink,$uid,$gid) = lstat($_)) &&
    -f _;
    if ($_ ne '.') { push @folders, "$user:$dir/$_"; }
}

sub rfc822 {

    my ($file) = @_;
    my ($rc) = 1;
    if (-d $file || -z $file || -B $file || -x $file) {
        $rc = 0;
    }
    return $rc;
}

sub rm_badchars {

    my ($mailbox) = @_;
    $mailbox =~ s/ /_040/g;
    $mailbox =~ s/\!/_041/g;
    $mailbox =~ s/\"/_042/g;
    $mailbox =~ s/\#/_043/g;
```

In this example, we omitted the remainder of the character translations, but in the actual script, each character that is not allowed should have a statement in the *rm_badchars* subroutine that translates it into its ASCII representation. After performing the translations, the new mailbox name is returned to the main program:

```
    return $mailbox;
}
```

createfolders

createfolders (shown in Example A-2) is a Tcl script that was used in Chapter 9 to create empty Cyrus mailboxes. It takes *bsd2cyrus*'s output as input. *bsd2cyrus*'s output contains the username, the pathname to a Berkeley-style mail folder, and

its mapping into the Cyrus namespace (i.e., the Cyrus mailbox name), but *createfolders* uses only the Cyrus mailbox name.

Example A-2. createfolders

```
#!/usr/local/bin/cyradm -file
set inputfile [lindex $argv 0]

eval cyradm connect cyr_conn localhost imap
puts stdout "Connected to IMAP server. Authenticating..."

if [catch {eval cyr_conn authenticate -pwcommand {{
    set adminid cyrusadm
    set adminpw xxxxxxxx
    list $adminid $adminpw
}} } result ] {
    puts stderr "$result (cleartext)"
    return -code error $result
} else {
    puts "Authentication successful."
}

## $inputfile is a text file containing username, path to
## Berkeley format folder, and corresponding Cyrus mailbox

if [catch {open $inputfile r} fileId] {
    puts stderr "Error: cannot open $inputfile"
} else {

    while {[gets $fileId line] >= 0} {

        ## The Cyrus mailbox is the second field in the input
        ## line (arrays are indexed starting with 0).

        set mailbox [lindex [split $line ":"] 1]

        if [catch {cyr_conn createmailbox $mailbox} result] {
            puts stderr $result
        } else {
            puts "Created mailbox $mailbox"
        }
    }
}
```

inboxfer

inboxfer is a Perl script that was used in Chapter 9 to move messages from a Berkeley format inbox (e.g., */var/mail/johndoe*) into a Cyrus inbox. The script (shown in Example A-3) assumes that the Cyrus mailbox already exists and that *formail* is

available on the machine where the script runs.* *inboxfer* takes the name of the file containing usernames, one per line, as input.

Example A-3. inboxfer

```
#!/usr/local/bin/perl

## Purpose: Extract messages from /var/mail mailbox
##          and populate the Cyrus INBOX.

$scripts   = "/home/cyrus/bin";       # Location of this script
$mailstore = "/var/spool/imap/user";  # Cyrus mailstore
$oldspool  = "/var/oldmail";          # Old mail spool

$cmd       = "/usr/local/bin/formail -n 20 -s $scripts/cpmsg";

$users = "$ARGV[0]";
if (!$users) { die "Usage: $0 $users\n"; }

open(USERS,"$users") || die "can't open $users";

while (<USERS>) {
    chop;
    $inbox = "$oldspool/$_";
    system("/usr/bin/cat $inbox | $cmd $mailstore/$_");
}
```

The system call pipes the contents of the incoming mail folder into a *formail* command. The *formail* command splits the folder up into separate mail messages and, in turn, pipes each mail message into another Perl script, *cpmsg*, for processing. *cpmsg* is the script that actually copies the message into the Cyrus mailbox. *cpmsg* is shown in Example A-4.

formail increments an environment variable, FILENO, each time it finds a new message in the Berkeley folder. *cpmsg* takes advantage of the FILENO variable to come up with a unique numbering scheme for the messages within a single Berkeley mail folder. The FILENO numbers are of the format NNN, padded with leading 0's. Because Cyrus works with ordinal numbers followed by a ".", *cpmsg* must remove leading 0's and add the "." to make a valid Cyrus message filename.

Do not use *inboxfer* to copy mail into Cyrus mailboxes that already contain mail!

* *formail* is part of the *procmail* distribution, available from *http://www.procmail.org*.

inboxfer always starts its numbering with "1.". If messages in the mailboxes have numbers between "1." and the number of messages being copied, they will be overwritten.

cpmsg takes the full path of the Cyrus mailbox as an argument (for example, the full path to *johndoe*'s INBOX on most Cyrus systems would be */var/spool/imap/user/johndoe*).

Example A-4. cpmsg

```
#!/usr/local/bin/perl

## This is to be called by formail. Formail calls this program
## once for each mail message when called with the -s option.
## E.g.:
##           cat mailbox.txt | formail -s thisscript.pl
##
## maildir - Directory where the mail message is to be written.

$maildir  = "$ARGV[0]";
if (!$maildir) { die "Usage: $0 $maildir"; }

## Formail increments this number for each message.
## The leading "0"'s must be removed (e.g. 001 becomes 1).

$filenum  = ($ENV{FILENO} - 0) + 1;

open (OUTFILE,">$maildir/$filenum.");
while (<STDIN>) {
    chop;
    print OUTFILE "$_\015\012";  ## Add CRLF to each line!
}
close OUTFILE;
```

folderxfer

folderxfer is very similar to *inboxfer* except that, instead of a list of usernames, it takes the output of *bsd2cyrus* as input and copies messages from Berkeley-format mail folders into the corresponding Cyrus mailboxes. The *folderxfer* script is shown in Example A-5.

Example A-5. folderxfer

```
#!/usr/local/bin/perl

## Purpose: Converts contents of Berkeley-format mail folders
##          to Cyrus mailboxes
##
## Assumptions: (1) The root mailbox and empty Cyrus folder must
##                  exist before conversion takes place.
##
```

Example A-5. folderxfer (continued)

```
##              (2) Input has been checked for illegal characters
##                  and files that do not contain mail content.
##
## Input:   A list containing the following information on each
##          line:
##
##      <username>:<cyrus-format folder name>:<BSD folder path>

$scripts     = "/home/cyrus/bin";       # Location of this script
$mailstore   = "/var/spool/imap/user";  # Cyrus mailstore
$cmd         = "/usr/local/bin/formail -n 20 -s $scripts/cpmsg";

$folders = "$ARGV[0]";
if (!$folders) { die "Usage: $0 filename"; }

open (MB,"$folders") || die "can't open $folders";
while (<MB>) {

    chop;

    ## Be careful with this split - the last token might have
    ## whitespace we want to preserve.

    ($user,$cyrusfolder,$folder) = split(/:/,$_,3);
    @fields = split(/\./,$cyrusfolder);
    $cyrfol = $fields[$#fields];

    $cat = "/usr/bin/cat \"$folder\"";
    system ("$cat | $cmd '$mailstore/$user/$cyrfol'");
}
close MB;
```

batchreconstruct

batchreconstruct is a Perl script that runs the Cyrus *reconstruct* command on each newly created Cyrus mailbox. It takes a filename or a list of usernames as input. The *batchreconstruct* script is shown in Example A-6.

Example A-6. batchreconstruct

```
#!/usr/local/bin/perl

chop ($whoami = `/usr/ucb/whoami`);
if ($whoami ne "cyrus" ) {
    die "You must be cyrus to run this script!\n";
}

$cmd   = "/usr/cyrus/bin/reconstruct -r";
$users = "$ARGV[0]";
if (!$users) { die "Usage: $0 input_file\n"; }
```

Example A-6. batchreconstruct (continued)

```
open(MB,"$users") || die "can't open $users";
while (<MB>) {
    chop;
    system("$cmd user.$_");
}
close MB;
```