AL_SCEPTOR(1L)

1. INPUT MESSAGE FORMAT

al_sceptor [-phw] <interceptee>

where the flags mean:
- p -> Don't print the messages being received.
- w -> Don't send watch dogs on to the destination.
- h -> Hold the message until manually released.

Valid Intercepted Processes and Abbreviations are:

ad	almdst
t1	ompr
a2	al2
m1	mmtpars
ac	acme
ab	almbld
tf	trfdc
d1	dataclerk

2. PURPOSE:

This command will intercept UNIX messages being sent to the named process. It is assumed that the messages correspond to the PIPE structure named in "/usr/include/alert.h". The contents of the message are displayed and the message then sent on to the destination process.

Options are available to eliminate watch dog messages and to hold messages until manually released (by typing a new-line char).

When a non-watchdog message is received, the screen is cleared, the contents of the message header and corresponding message are displayed, and the program sleeps for 5 seconds. As watch dogs are received, a message indicating reception is printed, the screen is not cleared. A sequence number is added to each watchdog for convenience of viewing.

EXAMPLE:

al_sceptor
will print help information for running the command.

al_sceptor ompr
will cause all messages being received by ompr to be displayed.

al_sceptor -w almdst
will cause all non-watchdog messages to be sent to the almdst.

al_sceptor -wh acme
will cause all non-watchdog messages to be sent to acme but each message received will only be sent on to acme after al_sceptor receives any new-line terminated input.

If the interceptee process is not running, al_sceptor will sit in a loop and try every 10 seconds to arouse that process. If the interceptee process should die while running al_sceptor, al_sceptor will again try every 10 seconds to talk to the destination.

A delete will terminate al_sceptor.

BUGS: Race conditions may occur while reading/setting process semaphores. When al_sceptor is run, the process id and semaphore number of the intercepted process is printed. When al_sceptor is terminated, the process id restored is also printed. If the process ids printed on entrance and exit are not the same, you may have to kill the intercepted process to restore sanity.

This procedure should rarely have to be used.

NAME

allofc - update/upgrade utility program

SYNPOSIS

allofc [**TYPE** ...] [**OPTION** [ARG ...] ...] command ...

DESCRIPTION

The allofc utility program was created to simplify the update/upgrade procedures. Allofc permits a sequence of UNIX commands to be performed in some or all office and type directories. The sequence of commands must all be UNIX commands which includes user programs, user shell files and SCCSH commands if they can be executed as UNIX commands. The allofc command line keywords (and arguments) are used to specify which commands to perform and which office or type directories the commands are performed in. These command line arguments and options are as follows:

TYPE is a string on one or more two-digit SPCS type numbers (e.g. **"01"** for No. 1 ESS; **"01 02"** or **"0102"** (the space is optional) for No. 1 ESS and No. 2 ESS). If one or more **TYPE**s are specified, then commands are executed only in office (or type for **"+t"** option) directories corresponding to these types. If no **TYPE** is specified, then commands are executed in all office (or type) directories.

OPTION is an option of the form **"+<option>"** or **"-<option>"**, possibly requiring one or more arguments (ARG). For the present the following **<options>** are available:

+e <filename>
The existance of the file, <filename>, is checked for before any commands are executed. If <filename> does not exist, then no commands are executed (i.e., only the directories in which <filename> exists will have commands executed in them).

+f all offices (or type) directories are finished despite any errors which may occur. If an error does occur, then allofc will report the error (including the directory in which it occurred) and the next office (or type) directory is tried. After completion the number of errors is printed, but only if one or more occurred.

+i All errors (abnormal terminations, systen errors, etc.) which occur while executing commands are ignored by allofc.

+sh <command>
Execute <command> using the shell and **"-c"** option (i.e., **"sh -c** <command>). This is only useful if errors from <command> are to be ignored.

+t Use type directories ("/type??") instead of office directories ("/office/<**name**>").

+u Append "/sccdev" to all directory names. This is used when the allofc command is part of the update/upgrade procedure.

-e <filename>
The existance of the file, <filename>, is checked for before any commands are executed. If <filename> exists, then no commands are executed (i.e., only the directories in which <filename> does not exist will have commands executed in them.

command is a UNIX command, optionally with arguments. If arguments are present the entire command must be enclosed in quotes (so that it appears to allofc as one argument). Allofc uses the execvp() subroutine described in see exec(2) when it executes a command. The order in which the commands appear in the allofc command line is the order in which the commands will be executed in a directory.

All "**+e**" options, "**-e**" options and **TYPE** specifications must be satisfied before any commands are executed in a particular directory. For example:

```
allofc 0107 -e .thresh +e .type/analth.proto \
    "cp .type/analth.proto .thresh"
```

For the allofc command above "**cp .type/analth.proto .thresh**" will be executed only in No. 1 ESS (01) and No. 1A (07) office directories in which ".thresh" does not exist and in which ".type/analth.proto" does exist.

SEE ALSO

exec(2) sh(1) perror(3)

DIAGNOSTICS

The diagnostics produced by allofc are intended to be self explanatory. In some cases perror(3) is used.

If allofc detects an error it will normally terminate. The use of the "**+sh** <command>" option will cause errors (non zero '**wait**()' values) which occur in <command> to be ignored by allofc. The use of the "**+i**" option will cause errors in any of the commands to be ignored.

BUGS

Any command which starts with a digit (**'0'-'9'**) is assumed to be a **"TYPE"** specification, so do not start any commands with digits. The correction of this bug would cause allofc to be unable to report syntax errors in a **"TYPE"** specification because it would not know which command line arguments were commands and which are **"TYPE"** specifications.

Allofc is very slow!

NAME

arcv - archive converter

SYNOPSIS

arcv afile [...]

DESCRIPTION

Arcv converts old-format archives into new-format archives. Arcv will not affect any file other than an old-format archive (first word contains 0177555). Appropriate comments are made while arcv works.

FILES

/tmp/arc???? temporary

SEE ALSO

archive(5)

NAME

audpr - Audit PR directory

SYNOPSIS

audpr [**-aensvy**] [alternate] name ...

DESCRIPTION

Audpr outputs the AU, GU, and PG files referenced by a given set of source files. The set of source files is given as an argument to audpr.

There are four ways, or modes, to express the arguments. Modes 1, 2, and 3 may be mixed as arguments; mode 4 arguments should not be mixed with the others. They are:

Mode 1 - name is an update directory on /genupd or **src**. If **src** is specified, /gensrc/src is assumed. Audpr assumes that the given directory contains pr directories. The set of source files includes all source files within the pr directories. An example of this mode is: **audpr u11**.

Mode 2 - name is an update directory on /genupd or **src** followed by a pr directory. The set of source files includes all files within the given pr directory. An example of this mode is: **audpr u10/pr-1p171**.

Mode 3 - name is an update directory on /genupd or **src** followed by a pr directory followed by a source file name. The set of source files is the given source file. An example of this mode is: **audpr src/pr-1p137/make02.c**.

Mode 4 - name is a simple source file name, not a partial or full path name. For example, make.c is okay, but pr-1p137/make.c is not. A mode 4 argument should not be mixed with arguments of modes 1, 2 or 3. Other simple source files may be specified. This mode should only be used when one's current directory is a pr directory within /gensrc/src or an update directory. Audpr will determine the pr directory and update directory from the names of the current directory's parent and grandparent. An example of this mode is: **audpr make.c**.

Audpr "knows" which update directories exist on /genupd, their sequence, and that /gensrc/src contains the latest "accepted" source for all generics. It knows that the first in the sequence of update directories (typically the lowest numbered update directory) will be incorporated into /gensrc/src before any other update directory. This means that a source file on an update directory is a more recent copy than and will eventually overwrite the corresponding older copy on /gensrc/src. Thus,

when given an update directory as an argument, audpr will use the latest copy of any data files residing on the update directory. The data files audpr must read to generate its output are AU, GU, and PG files.

As an example of how audpr examines update directories, assume there are two update directories, U1 and U2. U1 is sequenced first to be included into /gensrc/src. The process, **audpr U2** will examine first U2 for data (ie., AU, GU and PG files) then U1 and finally /gensrc/src. If there are copies of an AU file residing on U2, U1, and /gensrc/src, then audpr will process only the copy on U2 (the latest copy), ignoring the other two. The process, **audpr U1** will examine U1 first for data and then /gensrc/src. /gensrc/src is the last directory examined.

The flags available to audpr are:

a In its output, audpr normally abbreviates and combines the generic names of which a source file is a member. This flag inhibits this feature.

e Only errors are output. Error messages begin with an asterisk (*).

n A source file on an update directory but not on /gensrc/src (or alternate if specified) is denoted with a (**new**) message indicator when this flag is specified.

s Changes the default last directory examined from /gensrc/src to alternate. The alternate directory must be a full pathname and need not be a source directory on gensrc. However, audpr assumes it contains PR directories.

v causes audpr to verify the AU files it encounters in name. Audpr will print an error message on file descriptor 2 if it encounters an AU file satisfying one of the following:

1) The pident name in the NAME field differs from the name of the AU file, minus the .au.

2) The DOC field of the AU file specifies a PR number different from the actual PR directory the AU file resides in.

y The pr name normally output prior to processing a pr directory is inhibited. The pr name is output only while processing an update directory.

Audpr is designed to process efficiently pr directories with a large quantity of source files. As a result, the program may seem to run very slowly while processing a single source file.

FILES

/tmp/audprxxxx

SEE ALSO

sort(1), au(5L), gu(5L)

DIAGNOSTICS

BUGS

NAME

auex - execute command on source

SYNOPSIS

auex [**-acou**] file command-line

DESCRIPTION

Auex is a sophisticated program for sophisticated minds. In its simplest form it can be thought of as a program which reads pg, gu, and au files and executes a given command-line consisting of a unix command and arguments. The pg, gu and au files read can be specified as arguments via the use of special meaning keywords. The keywords which may be used and their special meaning are:

$pr	current pr number
$pg	current pg file name
$ve	version of the current gu file (assumes 2 charcaters)
$gu	current gu file name
$au	current au file name
$pr	current pr number

Auex, when given a pg file as the file argument, will remember the pg file name as the current pg file. The program then reads the pg file for the gu files listed therein. Only the #PR section is examined. Each gu file listed is processed one by one and is remembered as the current gu file as it is processed. The pr number of the gu file is also remembered as the current pr number. Each gu file may be read for the au files listed within it. Again, each au file, in turn, is remembered as the current au file name. Finally, each au file may be read and each source file therein remembered as the current source file name.

After all necessary readings are performed and the required special keywords evaluated, auex will execute the command-line. The actual execution is performed by executing the shell, sh (I), as a child and sending the command-line to the shell via a pipe.

Variations of the above theme follows. If the command-line contains no **$src** keyword, then no source file is processed. This actually avoids the need to read the au file and as a result, the program is faster because it does less work. Likewise, if no **$src** and no **$au** keywords are present in the command-line then the gu files are not read, saving more time. In short, auex reads only those source control files (pg, gu, and au) which are necessary to perform the keyword substitutions prior to executing the command-line.

Auex may be used to process a gu file (and all au and source files referenced by it) or an au file (and all source files referenced by it) by specifying the gu or au file as the file argument. In such a case undefined keywords such as **$pg** would have

a null value.

To process a pg file for the file argument, auex will chdir to the current pr directory before executing the command-line. As such, the program should be initially executed by a user whose current directory contains all the required pr directories (eg., /gensrc/src). To process a gu (or au) file, the user should reside in the directory with the au files and source files referenced by the gu file (or au file).

The flags available to auex:

- **-v** inhibit verbose output. Auex will execute the shell with the verbose option unless the **-v** option is specified.

- **-u** do not sort files. Auex will sort the au files listed in a gu file and the source files listed in an au file before processing them unless the **-u** option is specified.

- **-c** concatenate the source files listed in an au file into a single long string; each file name is separated by a blank. Backslash and newline characters are inserted between every fourth file name to make the string more readable by breaking it up into multi lines. The resulting string is used as the value for **$src**. If the keyword **$src** is not specified and the **-c** option is on, then the au files listed in a gu file are concatenated into one string and used for the value of **$au**.

- **-a** execute the command-line for all au files. Auex will normally not execute the command-line after processing an au file if 1) the command-line contains **$src** and 2) the au file has no source files listed in the #PROGRAM UNITS and #DATA UNITS sections. The **-a** option forces auex to execute the command-line for all au files, ignoring the above exception.

The quantity of arguments in the executed command-line may be large enough to exceed the shell's internal maximum number of arguments. This problem is most probable when the **-c** option is specified. Hence, a mechanism is included in auex to execute a command-line of at most 43 arguments when the **-c** option is specified. Thus, for a single au file several executions of the command-line may be required if the quantity of source files listed in the au file is large. In the following example, the au file, mine.au, has 50 source files:

```
auex -c mine.au 'ls $src'
```

and the ls command would be executed twice, the first time with 42 source files and the second time with 8 source files as the value of **$src**.

Auex must remember a lot and unfortunately it may run out of internal "memory". The only true solution is to recompile the program with more memory.

The character **$** has special meaning to auex. Besides interpreting **$pg**, **$pr**, **$gu**, **$au**, **$src** as special keywords, **$$** is mapped to **$** and $<anything else> is mapped to <anything else>.

FILES

SEE ALSO

au(5L), gu(5L), pg(5L)

DIAGNOSTICS

Error messages are to file descriptor two.

BUGS

Auex should dynamically allocate memory
A line read from a pg/gu/au file of greater than 200 characters may cause auex to drop a core

NAME

batch - run a process overnight

SYNOPSIS

batch <filename>

DESCRIPTION

The "batch" command is used to run large processes overnignt. The first (and only) argument is the name of a file containing a list of commands to be executed. Batch requests will be queued and then executed at 8:00 P.M. every evening. Output will be placed in the users home directory in a file named "bat#####.#.out". The starting and ending time for each batch process will be recorded in the file "batch.log" in the users home directory. In order to run "batch" the shell variables "$HOME" and "$PATH" must be defined.

FILES

/usr/bin/batch
/usr/bin/batchrun
/usr/lib/crontab
/etc/batchQ
/etc/bat.lck

BUGS

If two people attempt to enter a batch request at the same time - one user will get a "TRY AGAIN LATER" message.

NAME

bj - the game of black jack

SYNOPSIS

/usr/games/bj

DESCRIPTION

Bj is a serious attempt at simulating the dealer in the game of black jack (or twenty-one) as might be found in Reno. The following rules apply:

The bet is $2 every hand.

A player `natural' (black jack) pays $3. A dealer natural loses $2. Both dealer and player naturals is a `push' (no money exchange).

If the dealer has an ace up, the player is allowed to make an `insurance' bet against the chance of a dealer natural. If this bet is not taken, play resumes as normal. If the bet is taken, it is a side bet where the player wins $2 if the dealer has a natural and loses $1 if the dealer does not.

If the player is dealt two cards of the same value, he is allowed to `double'. He is allowed to play two hands, each with one of these cards. (The bet is doubled also; $2 on each hand.)

If a dealt hand has a total of ten or eleven, the player may `double down'. He may double the bet ($2 to $4) and receive exactly one more card on that hand.

Under normal play, the player may `hit' (draw a card) as long as his total is not over twenty-one. If the player `busts' (goes over twenty-one), the dealer wins the bet.

When the player `stands' (decides not to hit), the dealer hits until he attains a total of seventeen or more. If the dealer busts, the player wins the bet.

If both player and dealer stand, the one with the largest total wins. A tie is a push.

The machine deals and keeps score. The following questions will be asked at appropriate times. Each question is answered by y followed by a new line for `yes', or just new line for `no'.

```
?          (means, ``do you want a hit?'')
Insurance?
Double down?
```

Every time the deck is shuffled, the dealer so states and the `action' (total bet) and `standing' (total won or lost) is printed. To exit, hit the interrupt key (DEL) and the action and

standing will be printed.

BUGS

Be careful of the random number generator.

NAME

cdb - C debugger

SYNOPSIS

cdb [core [a.out]]

DESCRIPTION

Cdb is a debugging program for use with C programs. It is by no means completed, and this section is essentially only a placeholder for the actual description.

Even the present cdb has one useful feature: the command

$

will give a stack trace of the core image of a terminated C program. The calls are listed in the order made; the actual arguments to each routine are given in octal.

SEE ALSO

cc(1), C Reference Manual

BUGS

It has to be fixed to work with the new system.

NAME

chgmsg - modify SPCS output messages

SYNOPSIS

chgmsg [-s*n*] [-**D**] [-d*n*] [-c*n*] [-**N**] [-o*name*] [-f*x*,*y*,*z*] [-b*x*,*y*] *name*

DESCRIPTION

Chgmsg is a utility to modify SPCS output messages. It's features provide the ability to:

1. blank out n characters at the beginning of the original SPCS output message, where n is a number between 1 and 6 which specifies the "sort code" offset relative to the start of the original SPCS output message. The characters to be blanked out normally contain the "alarm class" and "time past hour" fields.

2. use the system date and time as the starting information to be placed in the message header for the first SPCS output message. The date and time elements for the message header of subsequent messages are incremented as necessary to simulate messages arriving in chronological order. The date and time elements are incremented as follows:

SECONDS	Current value plus 30 seconds.
MINUTES	Current value plus 5 minutes.
HOURS	Current value plus 1 hour.
DAYS	Current value plus 1 day.
MONTHS	Current value plus 1 month.

3. use the specified starting date and time, n, as the starting information for the message header of the first SPCS output message. The date and time elements for the message header of subsequent messages are incremented as described above in item 2. The permissible values of n range from 0101000000 to 1231235959.

4. change the channel ID element of the message header for all messages to n, where n is a number between 0 and 99.

5. change the message header delimiter for all messages to 0212.

6. redirect the modified messages to a specified output file.

7. change the contents of a specified field to a new value. If the length of the new field value is greater than the length of the old field value, then the new field value is

truncated to the length of the old field and copied to the message. When this happens, an appropriate error diagnostic is generated.

8. change all bytes having the value x to the new value y.

Normal output from chgmsg consists of a summary report on the standard error output device (file desc. 2) and a listing of all modified messages on the standard output device (file desc. 1); however, if the **-o** control option has been activated, then a listing of all modified messages is sent to the specified output file. The summary report indicates how many messages have been processed and prints a count of the number of times each requested control option (discussed below) has been invoked.

Chgmsg accepts a number of control options that specify how the messages are to be changed. A description of each control option follows:

-sn blanks out the "alarm class" and "time past hour" fields in the SPCS output message, where n specifies the "sort code" offset relative to the start of the original SPCS output message.

-D uses the system date and time as the starting information for the message header of the first message.

-dn uses the specified starting date and time for the message header of the first message, where the range of values for n is 0101000000 to 1231235959.

-cn uses the specified channel ID for all messages, where the range of values for n is 00 to 99.

-N changes the message header delimiter for all messages to 0212.

-oname redirects all modified messages to the specified output file, name. Please note that the summary report is not redirected.

-fx,y,z changes the specified field to the specified value, where x identifies the line (**0** to **GM_MAX_LNS**), y identifies the field to be changed, and z specifies the new value for the field. The value **GM_MAX_LNS** is defined in the header file, gtmhdr.h.

-bx,y changes all bytes having the specified "old" value, x, to the specified "new" value, y.

FILES

SEE ALSO

getfld(3L), gtmsg(3L)

DIAGNOSTICS

The following output messages are routed to the user's standard error output device (file descriptor= 2):

INVALID SORT CODE OFFSET. '-s' OPTION SKIPPED.

INVALID OPTION AND DATA FIELD, <opt>.

TIME OUT OF RANGE. '-d' OPTION SKIPPED.

DATE OUT OF RANGE. '-d' OPTION SKIPPED.

INVALID CHANNEL ID. '-c' OPTION SKIPPED.

OPTION <opt> HAS BAD DATA FIELD.

LINE NUMBER GREATER THAN GM_MAX_LNS. '-f' OPTION SKIPPED.

OLD VALUE OUT OF RANGE. '-b' OPTION SKIPPED.

NEW VALUE OUT OF RANGE. '-b' OPTION SKIPPED.

SKIPPING BAD OPTION, <opt>.

SYNTAX ERROR.

FILE <name> NONEXISTENT.

OLD FIELD SMALLER THAN NEW FIELD. NEW FIELD TRUNCATED TO <num> BYTES.

NAME

cmpl - compare with large block size

SYNOPSIS

cmpl [-l] file1 file2

DESCRIPTION

Cmpl compares two files, byte by byte.

Cmpl reads 10240 bytes per read. It will print out a check sum of both files whether an error has occurred or not. The check sum is to file descriptor 1 and is NOT the same check sum as that of the sum command. If a successful comparison has occurred, the word, identical, will be printed and a 0 error code return results. All other error code returns are 1. If the -l option is specified and the two files differ, the byte address (in decimal, starting with 0) and the differing bytes (in octal) are printed. Output is to file descriptor 1.

Error outputs (e.g., cannot open) are to file descriptor 2.

FILES

SEE ALSO

cmp(1)

DIAGNOSTICS

BUGS

Two. First, the byte address is an unsigned integer and can reach a maximum address of 65536. Second, the program will not correctly compare two tapes with different record sizes even if the sizes are less than to 10240.

NAME

cmplib - compare two libraries

SYNOPSIS

cmplib [**-cp**] lib1 lib2

DESCRIPTION

The two libraries are compared. The flags specify the types of differences for which cmplib looks. In any case, a list of differences between the two files is produced. The list consists of a flag followed by the mode, owner id, group id, size of the file, and a path name. The flags mean:

d (Delete) The named file is in lib1, but not in lib2.

c (Change) The named file is in both libraries; however, a difference between the files exists. If only the "p" option is in effect, the detected difference is in protection (i.e. mode, owner, or group). If only the "c" option is in effect (which is the default option), then the difference is in file content. Files have different content if the sizes are different, or if they are not equivalent in a byte-forbyte comparison. If both options are in effect, the difference is in protection or content.

a (Add) The named file exists in lib2, but does not exist in lib1.

If there are any file position differences, cmplib states that fact. There is a file position difference if there exists a pair of files such that: (i) both files exist in both libraries, and (ii) the relative position of the two files is reversed in the two libraries. Moreover, return code 0 is yielded for identical libraries, 1 for different libraries, and 2 for an inaccessible, missing, or otherwise improper argument.

SEE ALSO

archive(5)

BUGS