CiTR Unix Skills Answers

Questions are in multiple choice form, but there may be more than one correct answer to the questions. Tick (✔) all the answers that you think are correct. The rationale is that often in Unix there are more than one way to do a particular operation. Different skill levels are revealed by knowledge of mechanisms available to them.

Unix

Question 1: platforms and binary compatibility Intermediate

platforms and binary compatibility
[distinguish h/w and s/w are all intel/u$oft]

1 ❒ x.sh written under Sun Solaris on SPARC, run on Linux on Intel
1 In general shells scripts are compatible. Hardware dependence is achieved by the interpreter (sh) being native to each target platform.

2 ❒ x.o compiled on Linux on Intel, linked with main.o compiled on Windows on Intel, and if successful, run on Windows on Intel
1? .o file compatibility (of GNU). The .o file has only Intel instructions, and further assumption is that subroutine call sequences are the same.

3 ❒ x.o compiled on Sun/Solaris on Sparc, linked with main.o compiled on Sun/Solaris on Intel, and if successful, run on Sun/Solaris on Intel
-1 .o contains incompatible instructions.

4 ❒ x compiled and linked on Linux on Intel, run on Windows on Intel
-1 System call mapping table would differ for the systems. .exe format different from ELF.

5 ❒ x compiled and linked on Sun/Solaris on Sparc, run on Sun/Solaris on Intel
-1 contains incompatible instructions.

I really don't like this question, as I think it has too many interdependencies. FreeBSD may run Linux binaries (and vice versa?). Linux with WINE may run some windows. I have seen Unixes run binaries from other instruction sets, but invoking an emulation system, but I think these are too fiddly to say are general.

User

Question 2: Commands Beginner/Intermediate

1 ❒ `at` displays the ASCII table
-1 at runs a command at a specified time

2 ❒ `cat` joins files together
1 concatenates input files together and prints on stdout

3 ❒ `cmd` executes a new command shell
-1 cmd is not a standard unix command

4 ❒ `comm` finds common lines in input file
1 prints common lines in typically sorted input files

5 ❒ `creat` Makes a new file
-1 creat is a unix system call, but not a user command

6 ❒ `dirs` lists working directory stack
1 true for the bash shell[1]

7 ❒ `ksh` executes a new command shell
1 Executes the Korn shell

8 ❒ `lpq` will list files queued to the printer
1 line printer queue. Common form for BSD based systems.

1. is this too particular Unix specific. CiTR normally runs bash, and is available on most Unices.

9 ❒ `lpstat` will list files queued to the printer
1 line printer status. Common form if SystemV based.
10 ❒ `ls` lists files in folder/directory
1 display files in a directory
11 ❒ `mv` renames a file
1 moves a file
12 ❒ `pr` sends file to default printer
-1 filter which reformats input documents, but doesn't print them itself, this is done with 'lpr' or 'lp'.
13 ❒ `ps` sends file to default postscript printer
-1 displays unix processes running
14 ❒ `rm` deletes a file
1 removes a file
15 ❒ `tee` makes a second copy of input file
1 filter which also makes a copy of stdin to argument filename
16 ❒ `uniq` identifies duplicated files in a directory
-1 removes duplicated lines from a (sorted) file
17 ❒ `wc` display who is logged onto console
-1 word count
18 ❒ `write` sets the standard output filename
-1 sets up interactive dialogue with another user
19 ❒ `xargs` displays arguments used starting X server
-1 builds a command line based on names fed through stdin
20 ❒ `xdpyinfo` displays screen size
1 X display info, which includes screen dimensions

Question 3: directories/file naming Beginner/Intermediate
[basic file system understanding,
3.1 What is "`..`" in the file system?
1 ❒ all of the current subdirectories
-1
2 ❒ current root directory
-1
3 ❒ current working directory
-1
4 ❒ definition of a range of filenames
-1
5 ❒ device mounted in the filesystem
-1
6 ❒ parent directory
1 .. is a special directory entry pointing to the single parent directory that holds the current directory file. The root directory "/" is the one for which "."=="..". Special processing is included in the kernel to handle "chroot"-ed processes, which replaces the above test for processes for which chroot is not the root of the file system
3.2 Which of the following characters are legal in a unix file system's filename
1 ❒ Any alphanumeric character
1
2 ❒ Blank/space character
1 though it is a hassle for applications such as the shell

3 ❒ Back slash character "\"
1 though the shell typically needs special processing
4 ❒ Colon character ":"
1
5 ❒ Forward slash character "/"
-1 This is used by the filesystem when parsing a path specification to delimit directories, and hence can't be part of the filename

3.3 Which of the following is true about filenames in Unix
1 ❒ a period "." at the beginning of a filename indicates a directory
-1
2 ❒ a period "." at the beginning of a filename indicates a hidden file
1 ls ignores these files unless the "-a" for "all" option
3 ❒ a period "." designates file type and is hence not stored as part of the name
-1
4 ❒ a filename ending in ".a" is typically an object library
1 an archive
5 ❒ a filename ending in ".a" is typically an assembly language file
-1 sorry, ".s"
6 ❒ a filename ending in ".dll" is typically a shareable library
-1 sorry, that is Windows
7 ❒ a filename ending in ".lib" is typically an object library
-1 sorry.
8 ❒ a filename ending in ".s" is typically an assembly language file
1 usually only temporary files
9 ❒ a filename ending in ".so" is typically a shareable library
1 yes. Solaris use .so for shared libraries, Linux uses .so, Ultrix uses .so, FreeBSD uses .so.2.0, HP-UX uses .sl instead.
10 ❒ a filename ending in ".so" is typically a StarOffice document
-1

Question 4: File names Beginner

This tests filename regular expression name expansion by the shell.
[Unix vs. Dos naming [Ff][Rr]ed.[hc]]

4.1 `petty.c`
None of the file match!
1 ❒ `FReda.h`
-1
2 ❒ `Wilma.h`
-1
3 ❒ `Wilmah.h`
-1
4 ❒ `[bB]etty.c`
-1
5 ❒ `freda.c`
-1
6 ❒ `petty.h`
-1

4.2 `*.c`
1 ❒ `FReda.h`
-1

2 ❒ Wilma.h
-1
3 ❒ Wilmah.h
-1
4 ❒ [bB]etty.c
1
5 ❒ freda.c
1
6 ❒ petty.h
-1

4.3 *a.*

1 ❒ FReda.h
1
2 ❒ Wilma.h
1
3 ❒ Wilmah.h
-1
4 ❒ [bB]etty.c
-1
5 ❒ freda.c
1
6 ❒ petty.h
-1

4.4 [fF][rR]eda.*

1 ❒ FReda.h
1
2 ❒ Wilma.h
-1
3 ❒ Wilmah.h
-1
4 ❒ [bB]etty.c
-1
5 ❒ freda.c
1
6 ❒ petty.h
-1

4.5 [bB]etty.*

None match this. [bB]etty.c would match [bB]etty.c when executed from shell because if the globbing fails, it falls back to passing the unexpanded regular expression as an argument to the application (e.g. ls).

1 ❒ FReda.h
-1
2 ❒ Wilma.h
-1
3 ❒ Wilmah.h
-1
4 ❒ [bB]etty.c
-1
5 ❒ freda.c
-1

6 ❒ `petty.h`
-1

4.6 `[a-zA-Z]etty.*`

1 ❒ `FReda.h`
-1

2 ❒ `Wilma.h`
-1

3 ❒ `Wilmah.h`
-1

4 ❒ `[bB]etty.c`
-1

5 ❒ `freda.c`
-1

6 ❒ `petty.h`
1

Question 5: Editors Intermediate

1 ❒ using vi
1 correct: / finds first occurrence
nn finds next 2
cw changes the following word

2 ❒ using vi
-1 in vi mode, 3s changes the first 3 letters of Wilma.
in ex mode (which would be stretching the point anyway) ,
3 takes you to line 3 where the
substitute Changes Fred(Nurk) to Betty(Nurk). i.e. the wrong one

3 ❒ using notepad
-1 notepad isn't a Unix utility

4 ❒ using ex
1 /Fred finds first occurrence of "Fred"
/ / finds the next 2 lines with occurrences of the same string
s//Betty/ changes first occurrence on the line of same string

Question 6: filters Intermediate

[unix command line tools, e.g. tr, sed, sort?, wc]

1 ❒ # using strsub
-1 no such unix command

2 ❒ # using tr
1 correct

3 ❒ # using grep
-1 grep has no editing function

4 ❒ # using ed
1 correct[1]

Question 7: File permissions Beginner/Intermediate

7.1 What is displayed when you enter the command `more filea`

1 ❒ `filea`
-1

1. Is this too complex a form. The "\" is needed because of shell expansion. Alternate with sed?, awk? This tests understanding of 'here' documents too.

2 ❒ `filea: No such file or directory`
-1

3 ❒ `filea: Permission denied`
1 read for other is absent, stimpy isn't in staff group.

4 ❒ `Fri Sep 15 18:42:07 EST 2000`
-1

7.2 What is displayed when you enter the command `more dira/filec`

1 ❒ `filec`
-1

2 ❒ `filec: No such file or directory`
-1

3 ❒ `filec: Permission denied`
-1

4 ❒ `Fri Sep 15 18:42:07 EST 2000`
1 dira, while not readable is searchable, so by fully pathing the read permissions on the file take effect. You couldn't do an "ls dira"

7.3 What would be the intended purpose of the directory assign?

1 ❒ Place for a student group to collaborate on a project
-1 No read, so can't get other collaborators work out

2 ❒ Place for instructor to provide templates for students to fill in
-1 No read

3 ❒ Place for instructor to hide answers to a test
-1 Student can overwrite the instructors work.[1]

4 ❒ Place for students to submit assignment answers to a test
1 A student can put files in the directory, but not read what is in there. This is not a perfect solution because the name space for the directory is shared, and perhaps there is the ability to overwrite other student answers.

7.4 What is displayed when you enter the command `./id2`

1 ❒ `Fri Sep 15 18:42:07 EST 2000`
-1

2 ❒ `id2`
-1

3 ❒ `id2: Permission denied`
-1

4 ❒ `uid=987(stimpy) gid=1103(student)`
-1

5 ❒ `uid=987(stimpy) gid=1103(student) euid=370(bloggs)`
1 File is setuid, so executes with effective UID being the owner, bloggs.

6 ❒ `uid=987(stimpy) gid=1103(student) euid=987(stimpy)`
-1

7.5 You change your working directory to /tmp. Here you execute the command `date > now`[2]

1 ❒ `chmod 600 now`
-1

2 ❒ `umask 600`
-1

---

1. Should this be a 0 score instead?
2. This question is badly posed in that it is easy to guess at.

3 ❒ `umask 600 now`
-1

4 ❒ `umask 066`
1 More normally 077, but this will suffice.

5 ❒ `umask 066 now`
-1

Question 8: File duplication Intermediate

symbolic links hard vs. soft links and directory implication

8.1 `copy a b`

1 ❒ current date and time is displayed
-1 copy isn't a unix command

2 ❒ error message "no such file or directory"
1 copy isn't a unix command

8.2 `ln a b`

1 ❒ current date and time is displayed
1 ln creates a second directory entry for the file 'a'. Unix only removes the file when the link count drops to 0.

2 ❒ error message "no such file or directory"
-1

8.3 `ln -s a b`

1 ❒ current date and time is displayed
-1 symbolic links reference another file, so if original removed, then the reference is faulty

2 ❒ error message "no such file or directory"
1

8.4 `cat a b`

1 ❒ current date and time is displayed
-1 'cat' sends output to stdout, you need "> b" to create a copy in 'b'. As written the cat will fail with b: no such file or directory

2 ❒ error message "no such file or directory"
1

8.5 `cp a b`

1 ❒ current date and time is displayed
1 cp creates a whole new copy of the file, so the original can be removed without affecting the copy.

2 ❒ error message "no such file or directory"
-1

Question 9: kill intermediate

[difference between kill and kill -9 and kill -SIGUSR1]

9.1 `kill -9` process_id

1 ❒ lowers priority of process while system releases resources
-1

2 ❒ memory dump is left
-1

3 ❒ parent process is not sent an indication of process completion
-1

4 ❒ process is given a chance to clean up before exit
-1

5 ❒ process will always exit

1 9=SIGKILL is one of the signals that can't be caught by the kill-ed process, so no handler can be called to perform cleanup.

9.2 `kill -0` process_id

1 ❒ forces truncation of the memory dump file

-1

2 ❒ indicates whether process still exists

1 This asks the kernel to deliver a signal of 0 to the process. The kernel interprets this as a request to check if a signal could be delivered to the process, without actually doing so. Checks are made for process existence and privilege of the requesting process to deliver to that target process.

3 ❒ process and any child processes are forced to exit

-1

4 ❒ process will always exit

-1

9.3 `kill -SIGHUP` process_id

SIGHUP is typically delivered to processes that have a file open on a terminal interface for which the terminal driver believes there are modem control signals present. If the carrier signal is lost, then the process is delivered the hang up (SIGHUP).

1 ❒ ends a virtual machine session

-1

2 ❒ force a memory dump, without exiting process

-1

3 ❒ requests process to re-read configuration setups

1 Commonly used by Unix daemon processes for this purpose. Rather than a required action, this is common practice on Unix.

4 ❒ signify a processor halt, and process migration

-1

~~5 ❒ force process from remote computers to hang up. — "cu" does have this behavior~~

Question 10: X intermediate

10.1 Display a clock with the current time of brute on arch.

1 ❒ `xclock -display=brute`

-1

2 ❒ `xclock -display arch:0.0`

1 xclock runs as a client on brute. The display is screen 0 on arch

3 ❒ `xclock -xhost=arch`

-1

4 ❒ `xhost brute xclock=arch:0.0`

-1

10.2 On workstation arch, display a login window for wimp

1 ❒ `DISPLAY=arch:0.0; export DISPLAY xterm -e telnet wimp &`

1 xterm the client application runs on brute, displaying on screen 0 of arch, the shell running in that xterm execs the login (telnet) to wimp

2 ❒ `XHOST=arch; export XHOST xterm -e telnet wimp &`

-1

3 ❒ `XHOST=arch; export XHOST telnet wimp -e xterm &`

-1

4 ❒ `xhost wimp xterm=arch:0`
-1

5 ❒ `xterm -display arch:0 -e telnet wimp`
1 xterm the client application runs on brute, displaying on screen 0 of arch, the shell running in that xterm execs the login (telnet) to wimp.

6 ❒ `xterm -xhost=arch -e telnet wimp`
-1

Programmer

Question 11: compiling, linking, running Advanced

11.1 `ar -crv libocal.a a.o b.o c.o`

1 ❒ ar gives error “no such file or directory”
-1

2 ❒ ar gives error “duplicate symbol”
-1 b.c exports no symbols, so no clash

3 ❒ ar gives no error
1 b.c exports no symbols, so no clash

4 ❒ cc gives compilation error
-1

5 ❒ cc gives error “no such file or directory”
-1

6 ❒ cc gives error “duplicate symbol”
-1

7 ❒ cc gives error “missing symbol”
-1

8 ❒ a.out gives error “core dumped”
-1

9 ❒ a.out produces output “fb(fa())=%d”
-1

10 ❒ a.out produces output “fb(fa())=11”
-1

11 ❒ a.out produces output “fb(fa())=21”
-1

12 ❒ a.out produces no output
-1

11.2 `cc m.c a.o b.o ./a.out`

1 ❒ ar gives error “no such file or directory”
-1

2 ❒ ar gives error “duplicate symbol”
-1

3 ❒ ar gives no error
-1

4 ❒ cc gives compilation error
-1

5 ❒ cc gives error “no such file or directory”
-1

6 ❒ cc gives error “duplicate symbol”
-1

7 ❒ cc gives error “missing symbol”
1 b.o exports no symbols since all static, so ld phase of cc has m.o referencing the missing symbol “fb”
8 ❒ a.out gives error “core dumped”
-1
9 ❒ a.out produces output “fb(fa())=%d”
-1
10 ❒ a.out produces output “fb(fa())=11”
-1
11 ❒ a.out produces output “fb(fa())=21”
-1
12 ❒ a.out produces no output
-1

11.3 `cc m.c a.o c.o ./a.out`

1 ❒ ar gives error “no such file or directory”
-1
2 ❒ ar gives error “duplicate symbol”
-1
3 ❒ ar gives no error
-1
4 ❒ cc gives compilation error
-1
5 ❒ cc gives error “no such file or directory”
-1
6 ❒ cc gives error “duplicate symbol”
-1
7 ❒ cc gives error “missing symbol”
-1
8 ❒ a.out gives error “core dumped”
-1
9 ❒ a.out produces output “fb(fa())=%d”
-1
10 ❒ a.out produces output “fb(fa())=11”
-1
11 ❒ a.out produces output “fb(fa())=21”
1 everything compiles and runs
12 ❒ a.out produces no output
-1

11.4 `ar -crv libocal.a a.o c.o cc m.c -L. -local ./a.out`

1 ❒ ar gives error “no such file or directory”
-1
2 ❒ ar gives error “duplicate symbol”
-1
3 ❒ ar gives no error
0
4 ❒ cc gives compilation error
-1
5 ❒ cc gives error “no such file or directory”
-1

6 ❒ cc gives error “duplicate symbol”
-1
7 ❒ cc gives error “missing symbol”
-1
8 ❒ a.out gives error “core dumped”
-1
9 ❒ a.out produces output “fb(fa())=%d”
-1
10 ❒ a.out produces output “fb(fa())=11”
-1
11 ❒ a.out produces output “fb(fa())=21”
1 everything compiles and runs, a.o and c.o symbols come from library
12 ❒ a.out produces no output
-1

11.5 `ar -crv libocal.a a.o c.o cc m.c a.o c.o -L. -local ./a.out`

1 ❒ ar gives error “no such file or directory”
-1
2 ❒ ar gives error “duplicate symbol”
-1
4 ❒ cc gives compilation error
-1
5 ❒ cc gives error “no such file or directory”
-1
6 ❒ cc gives error “duplicate symbol”
-1
7 ❒ cc gives error “missing symbol”
-1
8 ❒ a.out gives error “core dumped”
-1
9 ❒ a.out produces output “fb(fa())=%d”
-1
10 ❒ a.out produces output “fb(fa())=11”
-1
11 ❒ a.out produces output “fb(fa())=21”
1 everything compiles and runs, a.o and c.o symbols come from a.o and c.o, the library is not needed or used since all symbols are resolved.
12 ❒ a.out produces no output
-1

11.6 `ar -crv libocal.a a.o c.o cc m.c -L. -local a.o c.o ./a.out`

1 ❒ ar gives error “no such file or directory”
-1
2 ❒ ar gives error “duplicate symbol”
-1
3 ❒ ar gives no error
0
4 ❒ cc gives compilation error
-1
5 ❒ cc gives error “no such file or directory”
-1

6 ❒ cc gives error “duplicate symbol”

1 symbols fa and fb are resolved from the library[1], and then clash with symbols in a.o and c.o on the command line and the global eleven also is duplicated

7 ❒ cc gives error “missing symbol”

-1

8 ❒ a.out gives error “core dumped”

-1

9 ❒ a.out produces output “fb(fa())=%d”

-1

10 ❒ a.out produces output “fb(fa())=11”

-1

11 ❒ a.out produces output “fb(fa())=21”

-1

12 ❒ a.out produces no output

-1

11.7 `ar -crv libocal.a a.o c.o cc n.c -L. -local ./a.out`

Note: There is a difference in Unix implementations yielding 2 correct answers to this question (**7** , **8**)

1 ❒ ar gives error “no such file or directory”

-1

2 ❒ ar gives error “duplicate symbol”

-1

3 ❒ ar gives no error

0

4 ❒ cc gives compilation error

-1 There is no common header file to give the compiler a clue as to the mismatch.

5 ❒ cc gives error “no such file or directory”

-1

6 ❒ cc gives error “duplicate symbol”

-1

7 ❒ cc gives error “missing symbol”

1 HP-UX’s object file format tags the symbol type as data or code, and so eleven the variable is different from eleven the function, and so detects this as a missing symbol.

8 ❒ a.out gives error “core dumped”

1 eleven is an exported symbol, and the incompatible C declarations are not picked up because of no common header file. There is no code to be found at ‘eleven’ and so leads to a core dump.

9 ❒ a.out produces output “fb(fa())=%d”

-1

10 ❒ a.out produces output “fb(fa())=11”

-1

11 ❒ a.out produces output “fb(fa())=21”

-1

1. While I haven’t found a counter example to this, there may be one. Perhaps lower the number of marks lost?

12 ❒ a.out produces no output
-1

Question 12: Libraries and system calls Intermediate/advanced
select [FDSET etc, waiting on multiple IOs, timeouts etc]

1 ❒ `calloc` is used to request initialized memory from the process heap
1 calloc() allocates space for an array initialized to zeros.

2 ❒ `chroot` is used to restrict the subtree of the file system a process sees
1 change root directory: the starting point for path searches for path names beginning with /

3 ❒ `exec` creates a new Unix process, and runs the given command
-1 NO, the exec family only do second part. Process creation is done with fork. This is a crucial Unix differentiator.

4 ❒ FD_ISSET can be used to determine when to read data from a remote computer
1 Yes. Determines is a file descriptor activity mask is set after returning from select.

5 ❒ `fcntl` can be used to set file locking
1 Yes, amongst many file control functions fcntl controls file record/segment locking.

6 ❒ `fopen` is used to read Unix directory files
-1 no. fopen is a buffered file open. This library delays disk I/O operations that would normally be initiated immediately by the open/read/write system calls. Unix directory files have complex structure and are queried through the opendir/dirent family of library calls.

7 ❒ `fork` is used to create a new directory entry for a file
-1 No. Fork creates a new process

8 ❒ `free` is used to remove a file from a directory
-1 returns a block of memory to the heap, such as allocated by calloc

9 ❒ `gethostbyaddr` reads machine location from installation log
-1 gethostbyaddr returns the host name, aliases of a node based on its IP address

10 ❒ `kill` sends a message code to another Unix process
1 send a signal to a process or a group of processes. The message being a signal code number.

11 ❒ `mkdir` is used to create a new directory
1 make a directory

12 ❒ `nice` is used by a process to change its running priority
1 change priority of a process

13 ❒ `open` is used to establish a handle to a file for subsequent reading
1 opens a file descriptor for the file for reading or writing

14 ❒ `fprintf` formats and prints arguments using a handle from `open`
-1 fprintf is part of the buffered IO family, the handle to this is a FILE* structure as returned by fopen. open returns a simple integer handle which is a key through process structures into the systems file system structures. The format and print functionality is correct.

15 ❒ `select` can be used to determine when to read data from a remote computer
1 select() examines the I/O file descriptor sets whose addresses are passed in readfds, writefds, and exceptfds to see if any of their file descriptors are ready for reading. Typically the file descriptors are for TCP connections to remote processes.

16 ❒ `signal` sends a message code to another Unix process
-1 signal sets up handlers for receiving signals from other processes, or for handling error conditions from within a process. kill signals can sometimes be caught by handlers set up by signal.

17 ❒ `socket` can be used to establish process to process communication
1 socket: create an endpoint for communication. Within a machine sockets can be created as Unix domain/named sockets or IP sockets.

18 ❒ `stat` can be used to find which user owns a file
1 stat: get file status. This returns file ownership, protection, size access dates etc.

19 ❒ `system` creates a new Unix process, and runs the given command
1 system - issue a shell command. This is a library call that handles the fork+exec+wait sequence of calls, and also required signal handling.

20 ❒ `unlink` is used to remove a file from a directory
1 unlink - remove directory entry. Strictly the file removal is a side effect of removing the last reference (directory entry or process file open).

Question 13: System utilities Advanced

Note: cd, ulimit, umask have shell script implementations as well to support 'exec'-ing of the command. These shell scripts simply invoke the builtin function with the arguments. Since the controlling shell is never returned to, it doesn't matter that the environment is lost.

13.1 CD Advanced

system/shell/process structure -- [e.g. environment variable visibility]

1 ❒ because the CD drive is viewed through the file system
-1

2 ❒ efficiency concerns
-1

3 ❒ faster as filesystem searching can be bypassed
-1

4 ❒ lack of a corresponding system call
-1 no chdir exists

5 ❒ it allows a history/audit trail to be maintained
0 It does allow a history, but this isn't a compelling reason.

6 ❒ it can only affect the currently running process
3 If "cd" were an application, then with the parent process being the shell, the fork-ed "cd" would execute the "chdir" call, and change the working directory of the child. The parent shell is unaffected by this, and hence when the "cd" terminates, nothing has changed.

13.2 Other special commands Advanced

1 ❒ `.`
1 Since environment variables are set in the interpreted script, these would be lost to the issuing shell if a subprocess were used.

2 ❒ `echo`
-1

3 ❒ `exec`
1 The side effect is that the controlling shell exits.

4 ❒ `find`
-1

5 ❒ `kill`
-1
6 ❒ `printenv`
-1
7 ❒ `time`
-1
8 ❒ `ulimit`
1 sets resource limits in the controlling shell, which will be inherited by subsequent children. If it were separate utility, the issuing shell would not get these set.
9 ❒ `umask`
1 sets file protection mask for file creation in the controlling shell, which will be inherited by subsequent children. If it were separate utility, the issuing shell would not get set.

Question 14: shared libraries Advanced
???

Question 15: Software development tools Intermediate
rcs/cvs/sccs experience

15.1 Automate a software build

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1
7 ❒ ldd
-1
8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
1
12 ❒ nm
-1
13 ❒ psrinfo
-1
14 ❒ RCS
-1
15 ❒ tar
-1

16 ❒ tr
-1

15.2 Coordinate file editing on a multi-developer/multi-file project

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1
7 ❒ ldd
-1
8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
-1
12 ❒ nm
-1
13 ❒ psrinfo
-1
14 ❒ RCS
1
15 ❒ tar
-1
16 ❒ tr
-1

15.3 Determine overall disk space utilization

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
1
6 ❒ du
-1
7 ❒ ldd
-1

8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
-1
12 ❒ nm
-1
13 ❒ psrinfo
-1
14 ❒ RCS
-1
15 ❒ tar
-1
16 ❒ tr
-1

15.4 Determine which symbols are referenced and exported by an object file

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1
7 ❒ ldd
-1
8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
-1
12 ❒ nm
1
13 ❒ psrinfo
-1
14 ❒ RCS
-1
15 ❒ tar
-1

16 ❒ tr
-1

15.5 Distributing a snapshot of a source tree to another office

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1
7 ❒ ldd
-1
8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
-1
12 ❒ nm
-1
13 ❒ psrinfo
-1
14 ❒ RCS
-1
15 ❒ tar
1
16 ❒ tr
-1

15.6 Examine which shared libraries are used by an executable

1 ❒ adb
-1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1

7 ❒ ldd

1 ldd is the tool for doing this on Solaris, FreeBSD and Linux. HP-UX uses chatr, but the output is considerably harder to interpret.

8 ❒ link

-1

9 ❒ lint

-1

10 ❒ lookbib

-1

11 ❒ make

-1

12 ❒ nm

-1

13 ❒ psrinfo

-1

14 ❒ RCS

-1

15 ❒ tar

-1

16 ❒ tr

-1

15.7 Regression test of trace output

1 ❒ adb

-1

2 ❒ ar

-1

3 ❒ builtin

-1

4 ❒ cmp

1 You would see if the output was different, but for a regression test the actual differences are not relevant. Diff is not given as an option anyway.

5 ❒ df

-1

6 ❒ du

-1

7 ❒ ldd

-1

8 ❒ link

-1

9 ❒ lint

-1

10 ❒ lookbib

-1

11 ❒ make

-1

12 ❒ nm

-1

13 ❒ psrinfo

-1

14 ❒ RCS
-1
15 ❒ tar
-1
16 ❒ tr
-1

15.8 Show a call traceback at the time of a program crash from its dump

1 ❒ adb
1
2 ❒ ar
-1
3 ❒ builtin
-1
4 ❒ cmp
-1
5 ❒ df
-1
6 ❒ du
-1
7 ❒ ldd
-1
8 ❒ link
-1
9 ❒ lint
-1
10 ❒ lookbib
-1
11 ❒ make
-1
12 ❒ nm
-1
13 ❒ psrinfo
-1
14 ❒ RCS
-1
15 ❒ tar
-1
16 ❒ tr
-1

Question 16: make Intermediate

1 ❒ `FRED=fred.c`
-1
2 ❒ `wilma: wilma.c`
-1
3 ❒ `cc -I. -o wilma wilma.c`
2 Most of the options given here are not shell commands, but makefile rules. This is the only compile line executed because we haven't specified a target on the command line, so the default is the first target of the file, wilma.

4 ❒ `fred: $(FRED)`
-1
5 ❒ `fred: fred.c`
-1
6 ❒ `cc -I. -o fred $(FRED)`
-1
7 ❒ `cc -I. -o fred fred.c`
-1
8 ❒ `flintstones: wilma fred`
-1
9 ❒ `wilma.c: flintstone.h`
-1
10 ❒ `$(FRED): flintstone.h`
-1
11 ❒ `fred.c: flintstone.h`
-1
12 ❒ `clean:`
-1
13 ❒ `rm Makefile wilma.c $(FRED) flintstone.h`
-1
15 ❒ `rm fred wilma`
-1

Documentation

Question 17: Unix documentation Intermediate

17.1 Unix `man` pages are prepared as

1 ❒ Latex files
-1 common tool but not for unix man command

2 ❒ Plain text files
-1 Sometimes systems cache man pages in this form, but original preparation is nroff. Sometimes formatted files are additionally distributed.

3 ❒ Rich text files
-1 common tool but not for unix man command. Sorry that's windows.

4 ❒ Nroff files
1 new-runoff. Using the "an" nroff macros

5 ❒ StarOffice documents
-1 common tool but not for unix man command

6 ❒ HTML files
-1 common tool but not for unix man command

7 ❒ Info files
-1 common tool but not for unix man command

8 ❒ Postscript files
-1 common tool but not for unix man command

17.2 Unix man pages are normally found in

1 ❒ /usr/man
1 yep, that's the traditional location

2 ❒ /opt/man
-1 no sorry

3 ❒ /usr/doc
-1 no sorry
4 ❒ /etc/man
-1 no sorry
5 ❒ /usr/local/emacs/manuals
-1 no sorry
6 ❒ /usr/share/man
1 yep, modern unix location

System Administrator

Question 18: Unix services and daemons intermediate

1 ❒ `at` executes recurrent user commands at specific times
-1 one-time job execution
2 ❒ `cron` executes recurrent user commands at specific times
1
3 ❒ `DNS` supports the digital certificate repository
-1 The Domain Name Service is implemented by the daemons - in.named, named, named-xfer - Internet domain name server
4 ❒ `inetd` starts IP protocol services
1
5 ❒ `init` adopts processes whose parent process has died
1
6 ❒ `init` starts specific other Unix services
1
7 ❒ `nfs` supports the network time synchronization service
-1 network file system
8 ❒ `NIS` includes a network wide login security database
1 network information service
9 ❒ `rlogin` provides restricted login
-1 remote login
10 ❒ `tftpd` services the trivial file transfer protocol
1 The listener daemon for the trivial file transfer protocol
11 ❒ `vmstat` displays operation of the Intel x86 emulation service
-1 virtual memory/paging/swapping statistics
~~**9** ❒ `sched` executes recurrent user commands at specific times~~
~~-1 kernel level scheduler~~
12 ❒ `YP` includes a network wide login security database
1 yellow pages. Original name for NIS until the phone book people invoked copyright.