Oscar Petersson, Matteus Laurent
oscpe262, matla782

TDDI41 Lab report

Högskoleingenjörsutbildning i datateknik, 180 hp

Laboration report - December 15, 2016
System Administration
TDDI41, Linköpings universitet

Assistant:
Jon Dybeck
IDA

LXB

Exercise 3: Navigating the Unix manual

3-1 Answer the following questions concerning sections of the Unix Manual. Most of the information you need can be found in the man page for the man command (you will need to think a little too).

a) Which are the nine sections of the Unix manual.

1	Executable programs or shell commands
2	System calls (functions provided by the kernel)
3	Library calls (functions within program libraries)
4	Special files (usually found in /dev)
5	File formats and conventions eg /etc/passwd
6	Games
7	Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8	System administration commands (usually only for root)
9	Kernel routines

b) Which section of the manual contains user commands such as cat and ls.

Section 1

c) Which section documents file formats, such as configuration files.

Section 5

d) Which section contains system administration commands, such as halt.

Section 8

4-1 The SYNOPSIS section briefly lists how a command is invoked. Read the man page for man and explain what the following command synopsis texts mean. Explain how each command may or must be invoked; you don't need to know what the commands actually do.

a) `mkpasswd PASSWORD SALT`

```
command argument argument
```

b) `uniq [OPTION]... [INPUT [OUTPUT]]`

```
command [runtime options goes here]...
[optional input stream[optional output stream]]
```

c) `gzip [-acdfhlLnNrtvV19 ] [-S suffix] [ name ... ]`

```
command [-available runtime flags w/o further arguments]
[-S flag needs argument (filename suffix)] [optional output name]
```

d)

```
chcon [OPTION]... CONTEXT FILE...
chcon [OPTION]... [-u USER] [-r ROLE] FILE...
chcon [OPTION]... --reference=RFILE FILE...
```

Three possible invokation syntaxes:

```
    command [runtime option] argument argument
    command [runtime option] [-u-flag argument] [-r-flag argument] argument
    command [runtime option] --reference-flag=argument argument
```

4-2 Read the man page for man, as well as some other man pages (e.g. for ls, uniq, chmod, and adduser) and answer the following questions.

a) What do you usually find in the DESCRIPTION section.

A brief briefing of what the entry is about, i.e. for an executable - what the executable' function is, and a description of flags etc.

b) Which section(s) usually document, in detail, what each command-line option does.

Section 1

c) Let's say you're reading the man page for a command and the information you are looking for isn't there. In which part can you find references to other man pages that might contain what you are looking for (have a look at the man page for reboot, imagining that you are trying to find a command that will turn the computer off, for an example).

Under **SEE ALSO**

d) In which section do you find information about which configuration files a program uses?

Section 1, usually a subsection

4-3 Use the man page for man to answer the following questions.

a) Sometimes there are several man pages (located in different sections) for the same keyword. Which command-line option to man can you use to display all of them.

```
man -a <search term>
```

b) Sometimes you don't know the name of a command you are looking for, but can guess at a keyword related to the command. Which command-line option can you use to have man list all pages related to a specific keyword.

```
man -k <keyword>
```

4-4 Display the man page for the ls command.

a) What does the ls command do.

It lists info about files in a given search path.

b) What option to ls shows information about file sizes, owner, group, permissions and so forth.

```
-l
```

c) What does the -R option to ls do? (Don't forget to try it.) Report: Brief written answers to the questions above.

Adds recursive directory search (lists subdirectories and their content as well).

Exercise 5: Absolute and relative path names

5-1 In the example above name at least one relative path name indicating ssh if:

a) The current working directory is /usr/bin.

```
./ssh/
```

```
../bin/ssh/
```

b) The current working directory is /usr/local/bin.

```
../../bin/ssh/
```

Exercise 6: Long format chmod

6-1 It is possible to set individual permissions for user, group and others using chmod. Review the documentation and answer the following questions.

a) How can you set the permission string to user read/write, group read, others read using chmod in long format.

```
# chmod a+rw-x,go-w <file>
```

b) How can you revoke group write permissions on a file without changing any other permissions.

```
# chmod g-w <file>
```

c) How can you grant user and group execute permissions without changing any other permissions.

```
# chmod ug+x foo
```

Exercise 7: Numeric file modes

7-1 What do the following numeric file modes represent.

a) 666

```
rw-rw-rw-
```

b) 770

```
rwxrwx---
```

c) 640

```
rw-r-----
```

d) 444

```
r--r--r--
```

7-2 What command-line argument to chmod allows you to alter the permissions of an entire directory tree.

```
-R, --recursive
```

7-3 What does execute (x) permission mean on directories.

Allows entering the directory and listing of its content.

7-4 A user wants to set the permissions of a directory tree rooted in dir so that the user and group can list, read and write (but not execute) files, but nobody else has any access. Which of the following commands is most appropriate? Why?

a) chmod -R 660 dir

b) chmod -R 770 dir

c) chmod -R u+rw,g+rw,o-rwx dir

(c) is the most suitable one, as it doesn't change possibly existing executive permissions for `[ug]`. *(a)* would remove any executive permissions on all files (and directories, making them inaccessible) for anyone, and *(b)* would add executive permissions for `[ug]` to all files and directories in the tree.

Exercise 8: Owner and group manipulation

8-1 How can you change the owner and group of an entire directory tree (a directory, its subdirectories and all the files they contain) with a single command.

```
# chown user:group -R <path root>
```

Exercise 9: File manipulation commands

9-1 What does cd .. do.

Changes current work directory to the parent directory.

9-2 What does cd ../.. do.

Changes current work directory to the parent directory of the parent directory.

9-3 What information about a file is shown by ls -laF?

Current directory content in the format: `filemode <> user group filesize(bytes) modification_time f`

9-4 In the following output from

```
ls -laFd dir dsp:
drwxr-xr-x 22 dave staff 4096   Jan 12 2001 dir/
crw-rw----  1 root audio 14, 3  Jan 22 2001 dspp
```

Explain the following:

a) What does the "c" at the beginning of the second line mean.

Character special file, e.g. pipes, ports, devices etc ...

b) What do "dave" and "staff" mean on the first line, and "root" and "audio" on the second.

User permissions for the files are set to the users "dave/root" and groups "root/audio" respectively.

c) Which users may write to the file dspp.

The user "root" and any users in the group "audio".

9-5 If you have two files, a and b, and you issue the command mv a b, what happens? Is there an option to mv that will issue a warning in this situation.

rename file 'a' to 'b'. `mv -i` will interactively ask for permission to overwrite.

9-6 What option(s) to cp allows you to copy the contents of /dir1 to /dir2, preserving modification times, ownership and permissions of all files.

```
$ cp -a /dir1/*.* /dir2/
```

9-7 How do you make the file secret readable and writable by root, readable by the group wheel and completely inaccessible to everybody else.

```
$ chown root:wheel secret && chmod 640 secret
```

Exercise 10: Shell init files

10-2 What init files does your shell use, and when are they used? (Hint: your shell has a man page, and somewhere near the end there is a section that lists which files it uses).

`/bin/bash`	The bash executable
`/etc/profile`	The systemwide initialization file, executed for login shells
`/etc/bash.bashrc`	The systemwide per-interactive-shell startup file
`/etc/bash.bash.logout`	The systemwide login shell cleanup file, executed when a login shell exits
`~/.bash_profile`	The personal initialization file, executed for login shells
`~/.bashrc`	The individual per-interactive-shell startup file
`~/.bash_logout`	The individual login shell cleanup file, executed when a login shell exits
`~/.inputrc`	Individual readline initialization file

Exercise 11: Manipulating environment variables

11-1 Use the env command to display all environment variables. What is PATH set to (you might want to use grep to find it)? What is this variable used for (the man pages for your shell might be helpful in answering this question).

```
PATH=/bin:/sbin:/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/bin/site_perl:\
/usr/bin/vendor_perl:/usr/bin/core_perl
```

`$PATH` is a list of paths from where the shell will recognize executables. Executables in `$PATH` can be executed from any search path without writing the full path to the executable.

11-2 Use echo to display the value of HOME. What does the HOME variable normally contain.

`$HOME` is the user "default" directory. Here the users document, user-specific configurations etc. are stored.

11-3 Prepend /data/kurs/adit/bin:/data/kurs/TDDI09/bin to the variable PATH. The easiest way to accomplish this is to use variable expansion in the right-hand side of the assignment.

```
$ PATH='/data/kurs/adit/bin:/data/kurs/TDDI41/bin:'${PATH}
```

Exercise 12: Redirecting output

12-1 Where will stdout and stderr be redirected in the following examples?

```
a) command >file1 b) command 2>&1 >file1 c) command >file1 2>&1
```

When answering these, remember that the order of redirections matters.

stdout:			stderr:		
	file1	console		file1	console
a	y	n	a	n	y
b	y	n	b	n	y
c	y	n	c	y	n

Exercise 13: Pipelines

13-1 What do the following commands do?

```
a) ls | grep -i doc
b) command 2>&1 | grep -i fail
c) command 2>&1 >/dev/null | grep -i fail
```

a) List directory content, select only the lines which contains the ordered character set "doc" ignoring upper/lower case.

b) Redirects stderr output of `command` to stdout which is then parsed for "fail" (ignoring u/l case)

c) As b, but will also put the stdout output into The Great Void.

13-2 Write command lines to perform the following tasks.

a) Output a recursive listing (using ls) of your home directory, including invisible files, to the file /tmp/HOMEFILES.

```
$ ls -aR ~ > /tmp/HOMEFILES
```

b) Find any files (using find) on the system that are world-writable (i.e. the write permission for "others" is set). Error messages should be discarded (redirected to /dev/null). This command is actually useful for auditing the security of a system - world-writable files can be security risks.

```
# find / -perm -o=w -type f 2>/dev/null
```

Exercise 14: Processes and jobs

14-1 Create a long running process by typing ping 127.0.0.1. Suspend it with vZ and bring it to the foreground with fg. Terminate it with vC.

14-2 Create a long running process in the background by typing

```
ping 127.0.0.1 >/dev/null&.
```

Find out its process id using ps and kill it using kill.

```
$ ping 127.0.0.1 >/dev/null&
kill $(ps -Ao pid=,comm --no-headers | grep ping | grep -o -E '[0-9]+')
```

(Or, just "ps", look for ping, send `kill <found PID>`), but that's a rather boring way, isn't it? ;)

14-3 What does the command kill -9 pid do, where pid is the number of a process? What does kill -9 -1 do? Read the documentation to figure the last one out as it is a somewhat dangerous command.

```
$ kill -9 <pid>
```

This line sends **SIGKILL** to the process with given PID. This can not be caught or ignored by the process.

```
$ kill -9 -1
```

This line sends **SIGKILL** to all processes with a PID larger than 1, i.e. every process apart from the init process started by the boot manager.

14-4 Create a long running process in the background. Kill it using pkill. The pkill command is very useful when you need to kill several processes that share some attribute (such as a command name).

```
$ pkill ping
```

Exercise 16: Using the pager less

16-1 What keystroke in less moves to the beginning of the file.

16-2 What keystroke in less moves to the end of the file.

16-3 What would you type in less to start searching for "option".

16-4 What would you type in less to move to the next match for "option".

BOF: g EOF: G forward search: /option backward search: ?option

16-5 Locate the package documentation for the ssh package and answer the following questions by reading the README.Debian.gz file (hint: remembering the answers to these questions may be useful in the project).

a) What is the default setting for ForwardX11.

For security reasons, `X11Forwarding` is disabled by default, this goes for Debian's `openssh-server` package as well.

b) If you want X11 forwarding to work, what other package(s) need to be installed on the server.

`xauth` at a bare minimum, plus of course any related packages for whatever functionality desired.

Exercise 17: Using non-interactive text editors (this exercise is optional)

17-1 Use sed to change all occurrences of "/bin/tcsh" to "/bin/sh" in /etc/passwd (output to a different file). This exercise is optional.

```
# sed -i "s|bin/tcsh|bin/sh|g" < /etc/passwd > foo
```

17-2 Examine the files shadow and passwd in the directory /data/kurs/TDDI09/labs/lxb. Use paste and awk to output a file where each line consists of column one from passwd and column two from the corresponding line in shadow. The printf function in awk is helpful here. This exercise is optional since it goes beyond the basics.

```
$ paste <(awk -F ":" '{print $2}' passwd ) <(awk -F ":" '{print $3}' shadow)
```

Exercise 18: Log files

18-1 What does tail -f /var/log/syslog do.

Prints the last 10 lines of `/var/log/syslog` and keeps parsing.

18-2 If you want to extract the last ten lines in /var/log/syslog that are related to the service cron, what command would you use? (Hint: the grep command can search for matching lines in a file).

```
$ cat /var/log/syslog | grep cron
```

Part 5 Errata:

The default configuration for most contemporary distributions is systemd, regardless of what one thinks of that. Sure, sysvinit might be a reasonable (at least you can reason wether it is or not ...) choice regarding this course, but that does not change the fact that the major distributions likely to be found in a professional environment (Debian, Fedora, SUSE, Red Hat, buntu, Mint)- except for legacy systems - are not running sysvinit.

Exercise 19: Booting Linux

19-1 What services are started when your system boots into run level 2 (i.e. not only services exclusive to run level 2, but all services started as the computer boots into run level 2).

bootlogs, syslog, ssh, cron, rmnologin, stopbootlogs

19-2 If you wanted to restart the ssh process, what command would you use.

```
$ systemctl restart sshd.service
-bash: systemctl: command not found
```

Oh right, old habits die hard ... :p

```
$ /etc/init.d/ssh restart
```

Exercise 20: Mostly hard stuff

20-1 Where will stdout and stderr be redirected in the following examples:

`a) command 2>&1 >file1 b) command >file1 2>&1` *see 12-1 b/c*

20-2 Using only ps, grep and kill, write a command that kills any process with "linux" in the name. Note that you must avoid killing the grep process itself (this can be done by crafting the regexp appropriately)! This command line can be used to rapidly terminate any UML running on the system.

```
$ kill $(ps -Ao pid,comm --no-headers | grep ping | grep -o -E '[0-9]+')
```

20-3 Use your shell's alias or function features to create a shell command named gzless that decompresses a gzip-compressed file and pipes it into less.

```
# ~/.aliases
gzless() {
  gunzip -c $1 | less
}
```

20-4 I/O redirection in the shell is somewhat limited. In particular, you can't handle programs that write directly to the terminal device (i.e. use a pseudo-terminal) rather than use the file descriptors. For situations like that, there's socat. Since you're an advanced user, you really should learn about socat.

a) Use socat so you can log in and execute commands on a host non-interactively using the password authentication method. Essentially, you should be able to do the equivalent of

```
 cat commandfile | ssh user@remote.host
```

, where commandfile contains the stuff you would normally type interactively.

```
$ file=$(cat commandfile) && \
  (sleep 3; echo PASSWD; sleep 1; echo "$file"; sleep 1) |\
  socat - EXEC:'ssh -l USER remote.ip.addr',pty,setsid,ctty
```

b) Do something else with socat that's neat. Come up with something on your own, perhaps using the network.

20-5 Write a command that renames a bunch of files to be all lowercase (e.g. BOFH.GIF is renamed to bofh.gif). You do not have to worry about spaces (but you should, just on general principle). The tr command may be useful here.

```
#!/bin/bash

for OLD in `find $1 -depth`
do
   NEW=`dirname "${OLD}"`/`basename "${OLD}" | tr '[A-Z ]' '[a-z ]'`
   if [ "${OLD}" != "${NEW}" ]
   then
[ ! -e "${NEW}" ] && mv -T "${OLD}" "${NEW}" \
|| echo "Could not rename ${OLD}. ${NEW} already exists."
   fi
done
```

20-6 Examine the files /data/kurs/TDDI09/labs/lxb/passwd and shadow (same directory). Use paste and awk to output a file where each line consists of column one from passwd and column two from the corresponding line in shadow. The printf function in awk may be helpful here.

see 17-2

20-7 Sometimes you need to change the IP address of a computer. Assuming that all the files that need to be changed are somewhere in /etc (or a subdirectory thereof), what command will list all relevant files and not print any error messages.

`# find ./ -type f | xargs grep "$oldip" 2>/dev/null` ...where `$oldip` is the IP-address we are looking for, of course.

20-8 Write a command line that changes the IP address stored in the variable (shell or environment) OLD to the IP address stored in the variable NEW in all (regular) files in /etc or its subdirectories. The commands find, xargs and sed may be useful here.

```
# find /etc -type f -exec sed -i "s|$OLD|$NEW|g" {} \;
```

20-9 Write a command that uses ssh to log in to all computers w0001 through w2000 and runs the w command on each one. You might want to read about arithmetic substitution in bash for this one. How could you avoid having ssh ask for a password for each computer without setting an empty password or resorting to host-based authentication.

Avoiding password login would be done through pubkey-based access:

```
#/etc/ssh/sshd_config
 PasswordAuthentication no
 PermitEmptyPasswords no
 [...]
#EOF
```

Generating and importing keys ... `ssh-keygen`

We could use `sshpass -p <PASSWD> ssh-copy-id -i ~/.ssh/<id-file> <user>@<host>` to avoid manually entering the passwords, at least if they are identical on all hosts...

Once sorted (phew) we'll simply run:

```
$ for client in ‘seq 0001 2000‘
  do
    ssh user@$client <<FOOXECUTETHIS
    w
    FOOXECUTETHIS
  done
```

Exercise 21: Quite hard stuff (optional)

21-1 Write a command that lists, for the last ten unique users to log in to the system, the last time they logged in using ssh (users can be found using last, and ssh logins in /var/log/auth.log, which is typically only readable by root).

21-2 Explain the following fairly contrived code, in particular all the I/O redirections.

The code starts out with directing two file descriptors, one for input and one for output (l.2). We look for .bak-files throughout the file system and print their path on a line each (l.3). We read these lines (l.5), ask (l.6) the user to confirm that we should log the entry to `/tmp/RECORD (l.2)`. If (l.8) the user answers (l.7) ”y”, we log the entry (l.2).

21-3 For each file on the system with a .bak suffix, where there either is no corresponding file without the .bak suffix, or the corresponding file is older than the .bak file, ask the user whether to rename the .bak file to remove the suffix. If the answer begins with Y or y, rename the file appropriately. You must handle file names containing single spaces correctly.

```
#/bin/bash
exec 23<&0 24>&1 <<EOF
‘find . -name "*.bak" -print‘
EOF
while read BAK; do
    FILE=$(echo "$BAK" | sed ’s|\.bak$||g’)
    if [ ! -e "$FILE" ] || [ "${BAK}" -nt "${FILE}" ]; then
       echo -n "$FILE not found or older than $BAK. Replace $FILE? Y/N"
       read ans <&23
       if [ "$ans" = "y" ] || [ "$ans" = "Y" ]; then
  mv -T "$BAK" "$FILE"
       fi
    fi
done
```