

Oscar Petersson, Matteus Laurent  
oscpe262, matla782

## **TDDI41 Lab report**

Högskoleingenjörsutbildning i datateknik, 180 hp

Laboration report - September 29, 2016  
**System Administration**  
TDDI41, Linköpings universitet

Assistant:  
Jon Dybeck  
IDA

# LXB

## 3-1

a)

- 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) Section 1

c) Section 5

d) Section 8

## 4-1

a)

`command argument argument`

b)

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

c)

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

d) 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

a) 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) Section 1

c) Under **SEE ALSO**

d) Section 1, usually a subsection

### 4-3

a) `man -a <search term>`

b) `man -k <keyword>`

### 4-4

a) It lists info about files in a given search path.

b) `-l`

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

### 5-1

a) `./ssh/`

`../bin/ssh/`

b) `../../bin/ssh/`

### 6-1

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

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

c) `# chmod ug+x foo`

### 7-1

a) `rw-rw-rw-`

b) `rw-rwx---`

c) `rw-r-----`

d) `r--r--r--`

### 7-2

`-R, --recursive`

### 7-3

Allows entering the directory and listing of its content.

## 7-4

(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.

## 8-1

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

## 9-1,2,3

1) Changes current work directory to the parent directory.

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

3) Current directory content in the format:

```
filemode <> user group filesize(bytes) modification_time filename[entry type indicator]
```

## 9-4

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

## 9-5

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

## 9-6

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

## 9-7

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

## 10-2

|                       |  |
|-----------------------|--|
| /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                                    |

## 11-1

```
PATH=/bin:/sbin:/usr/local/sbin:/usr/local/bin:/usr/bin:/usr/bin/site_perl:\n/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

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

## 11-3

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

## 12-1

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

## 13-1

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

- a) `$ ls -aR ~ > /tmp/HOMEFILES`
- b) `# find / -perm -o=w -type f 2>/dev/null`

## 14-2

```
$ 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

```
$ 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

```
$ pkill ping
```

## 16-1,2,3,4

BOF: g

EOF: G

forward search: /option

backward search: ?option

## 16-5

- a) For security reasons, **X11Forwarding** is disabled by default, this goes for Debian's **openssh-server** package as well.
- b) **xauth** at a bare minimum, plus of course any related packages for whatever functionality desired.

## 17-1 (optional)

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

## 17-2 (optional)

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

## 18-1

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

## 18-2

```
$ 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 whether 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`.

## 19-1

`bootlogs`, `syslog`, `ssh`, `cron`, `rmnologin`, `stopbootlogs`

## 19-2

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

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

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

## 20-1

*see 12-1 b/c*

## 20-2

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

## 20-3

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

## 20-4

a)

```
$ 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)

## 20-5

```
#!/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

*see 17-2*

## 20-7

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

## 20-8

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



## 20-9

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

```
#!/etc/ssh/sshd_config
PasswordAuthentication no
PermitEmptyPasswords no
RSAAuthentication yes
DSAAAuthentication yes
HostKey <key path>
[...]
#EOF
```

Generating and importing keys ...

Once sorted (pew), and supposing we have the computers defined in `/etc/hosts`, we'll simply run

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

## 21-1

## 21-2

The code starts out with directing two file descriptors, one for input and one for output (1.2). We look for .bak-files throughout the file system and print their path on a line each (1.3). We read these lines (1.5), ask (1.6) the user to confirm that we should log the entry to `/tmp/RECORD` (1.2). If (1.8) the user answers (1.7) "y", we log the entry (1.2).

## 21-3

```
#!/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
```