${Unix_Tools} –

exercises with some example solutions for supervisors

Markus Kuhn

Michaelmas 2004 – Part Ib

http://www.cl.cam.ac.uk/Teaching/2004/UnixTools/

Exercise 1 Write a shell command line that appends `:/usr/X11R6/man` to the end of the environment variable `$MANPATH`.

Example solution:

```
$ MANPATH="$MANPATH:/usr/X11/man"
```

Exercise 2 Create a new subdirectory and in it five files with unusual filenames that someone unfamiliar with the shell will find difficult to remove. Ask a fellow student to write down for each file the command line that will remove it.

Example solution:

```
$ mkdir /tmp/odd_names ; cd /tmp/odd_names
$ touch -- -i ' ' "\t" 'test.txt ' '...' $(echo -e "\177\177\177")
```

Exercise 3 Given a large set of daily logfiles with date-dependent names of the form `log.`*yyyymmdd*, write down the shortest possible command line that concatenates all files from 1 October 1999 to 7 July 2002 into a single file `archive` in chronological order.

Example solution:

```
$ cat log.19991??? log.200[01]???? log.20020[1-6]?? log.2002070[1-7] >archive
```

Exercise 4 Write down the command line that appends the current date and time (in Universal Time) and the Internet name of the current host to the logfile for the respective current day (local time), using the above logfile naming convention.

Example solution:

```
$ date -u >>log.$(date +'%Y%m%d') ; hostname >>log.$(date +'%Y%m%d')
```

or

```
$ { date -u ; hostname ; } >>log.$(date +'%Y%m%d')
```

Exercise 5 Configure your PWF-Linux account, such that each time you log in, an email gets sent automatically to your Hermes mailbox. It should contain in the subject line the name of the machine on which the reported login took place, as well as the time of day. In the message body, you should add a greeting followed by the output of the "`w`" command that shows who else is currently using this machine.

Example solution:

Add the following line to one of your startup scripts:

```
{ echo 'Hello there!' ; w ; } | \
  mail ${LOGNAME}@cam.ac.uk -s "Login on `hostname` on `date`"
```

Add this line to the file '`.bash_profile`', which will be executed whenever you log in via the text console (press Alt-Ctrl-F1) or remotely (e.g. `ssh linux2.pwf.cl.cam.ac.uk`). Where to add it such that it gets executed when you log in via the X Window System depends a lot on the specific installation that you use (usually you have to use `~/.xsession`).
[The backslash at the end of the first line says that this command line continues in the next line.]

Exercise 6 Explain what happens if the command "`rm *`" is executed in a subdirectory that contains a file named "`-i`".

Example solution:

The file '`-i`' is placed by the pathname expansion as one of the first words onto the command line. The rm command will recognize it as the command line option that asks for the interactive confirmation of each filename before it is deleted. (Some people place an empty '`-i`' file in important directories as a safeguard against an accidentally executed '`rm *`')

Exercise 7 Write a shell script "`start_terminal`" that starts a new "`xterm`" process and appends its process ID to the file `~/.terminal.pids`. If the environment variable `$TERMINAL` has a value, then its content shall name the command to be started instead of "`xterm`".

Example solution:

```
#!/bin/bash
if [ "$TERMINAL" != '' ] ; then
  $TERMINAL &
else
  xterm &
fi
echo $! >>~/.terminal.pids
```

Exercise 8 Write a further shell script "`kill_terminals`" that sends a SIGINT signal to all the processes listed in the file generated in the previous exercise (if it exists) and removes it afterwards.

Example solution:

```
#!/bin/bash
if [ -f ~/.terminal.pids ] ; then
  for i in `cat ~/.terminal.pids` ; do
    kill $i
  done
  rm ~/.terminal.pids
fi
```

Exercise 9 Write down the command line of the single `sed` invocation that performs the same action as the pipe

```
head -n 12 <input | tail -n 7 | grep 'with'
```

Example solution:

```
$ sed -e '6,12!d' -e '/with/!d' input
```

Exercise 10 Generate a CVS repository and place all your exercise solution files created so far into it. Then modify a file, commit the change, and create a patch file that contains the modification you made. And finally, retrieve the original version of the modified file again out of the repository.

Exercise 11 Add a `Makefile` with a target `solutions.tar.gz` that packs up all your solutions files into a compressed archive file. Ensure that calling `make solutions.tar.gz` will recreate the compressed package only after you have actually modified one of the files in the package.

Exercise 12 Write a C program that divides a variable by zero and execute it. Use `gdb` to determine from the resulting `core` file the line number in which the division occurred and the value of the variable involved.

Example solution:

```
$ ulimit -S -c unlimited            # enable generation of core files
$ cat >t.c <<EOF
#include <stdio.h>
int main(void) {
  int a = 0;
  a = 1/a;
  printf("a = %d\n", a);
  return 0;
}
EOF
$ gcc -ggdb -o t t.c
$ ./t
Floating point exception (core dumped)
$ gdb t core
[...]
Core was generated by `./t'.
Program terminated with signal 8, Arithmetic exception.
[...]
#0  0x08048347 in main () at t.c:4
4         a = 1/a;
(gdb) p a
$1 = 0
(gdb) q
```

Exercise 13 When editing sentences, users of text editors occasionally leave some word duplicated by by accident. Write a Perl script that reads plain text files and outputs all their lines that contain the same word twice in a row. Extend your program to detect also the cases where the two occurrences of the same word are separated by a line feed.

Example solution:

```
#!/usr/bin/perl
# spot consecutive repetition of the same word
while (<>) {
    # split line into array of words, separated by space or punctuation
    @words = split /[\s.,\'\`\(\)\[\]\{\}]+/;
    # compare neighboring words
    for ($i = 0; $i < $#words; $i++) {
        if ($words[$i] eq $words[$i+1]) {
            print "$_ => double '$words[$i]'\n";
        }
    }
    # compare first word of this line with last word of previous line
    if ($words[0] eq $prevword) {
        print "$prevline$_ => double '$prevword'\n";
    }
    $prevword = $words[$#words];
    $prevline = $_;
}
```

Exercise 14 Type in the file `example.tex` on slide 95. Call "`latex example`" twice. Preview with "`xdvi example`" the formatted text in the device-independent format (DVI) and convert it with "`dvips -Ppdf example`" to PostScript. View with "`ghostview example.ps`" and convert with "`ps2pdf example.ps`" into the *Portable Document Format*. Finally, call "`acroread example.pdf &`" to inspect the end of this text-format odyssey.

Exercise 15 Read pages 1–64 of the LaTeX book, then write your CV with LaTeX, convert the result into PDF, and put it onto your PWF homepage.
See `http://www.cam.ac.uk/cs/pwf/web/` for information on how to set up a homepage under PWF Linux.

Exercise 16 In a job interview for a position as a subeditor of a technical journal, your skills in spotting typographic mistakes made by LaTeX beginners are tested with this example text:

> The -7 dB loss ($\pm 2dB$) shown on pp. 7-9 can be attributed to the f(t)= $sin(2\pi ft)$signal , where t is the the time and f =48Khz is the "sampling frequency".

Can you spot all 14 mistakes? Write down both the probable original incorrect LaTeX source text, as well as a corrected version.

Example solution:

Original LaTeX source:

```
The -7 dB loss ($\pm 2 dB$) shown on pp. 7-9 can be attributed
to the f(t)$=sin(2\pi ft)$signal , where t is the the time and
$f=$48Khz is the ''sampling frequency''.
```

Corrected version:

```
The $-7$~dB loss ($\pm 2$~dB) shown on pp.~7--9 can be attributed
to the $f(t)=\sin(2\pi ft)$ signal, where $t$ is the time and
$f=48$~kHz is the ``sampling frequency''.
```

Corrected output:

> The −7 dB loss (±2 dB) shown on pp. 7–9 can be attributed to the $f(t) = \sin(2\pi ft)$ signal, where t is the time and $f = 48$ kHz is the "sampling frequency".

- Use math mode to typeset the minus sign, otherwise it becomes the much shorter hyphen.
  -7 → −7
- Units of measurement are typeset outside math mode in an upright font. Math-mode italic is used only for single-letter variables, not for units or constants.
  $\pm 2dB$ → ±2 dB
- The full-stop after pp will be treated as a wider end-of-sentence space by LaTeX, unless it is escaped with a backslash or replaced with a no-break space (~).
  pp. 7 → pp. 7
- Number ranges are traditionally written with an en-dash, not with a hyphen.
  7-9 → 7–9
- Since sin is a constant function, it has to be set in an upright font.
  $sin()$ → sin()
- Any binary mathematical operator or relation must have both sides set in math mode, otherwise the spaces that math mode inserts on either side of them will not appear correctly.
  $f =48$ → $f = 48$
- Each occurance of a variable must be set in math mode.
  t → t
- Symbols for SI units have a standardized usage of uppercase versus lowercase letters. (Prefixes kilo and lower are abbreviated with a lowercase letter, mega and above with an uppercase letter. The symbols of base units named after a person start with an uppercase letter.)
  Khz → kHz
- SI units are separated by a space from the preceding number (ISO 31).
  48kHz → 48 kHz
- Punctuation characters . , ; : ! ?) ’ ” are followed – but not preceded – by a space, while ‘ “ (are preceded – but not followed – by one.
)signal , →) signal,

Since π is a constant, it should, in principle, not be set in a math italic font, either. Unfortunately, LaTeX lacks by default upright Greek characters, therefore this is widely ignored.