${Unix_Tools}
– exercises

Markus Kuhn

Computer Science Tripos – Part IB

1 The shell

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

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.

[Your MCS Linux home directory sits on a Windows NTFS/CIFS file server, which does not use a Unix-style file system and has many additional restrictions on filenames. Therefore, better perform this experiment in a local subdirectory, which you can create with "`mkdir /tmp/$USER-odd-names/`".]

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.

You can fill a directory with 1000 test files using the following `make-logs.pl` Perl script:

```
#!/usr/bin/perl
use POSIX;
for $i (1..1000) {
  $t=rand 60*60*24*365*45;
  $f=strftime("log.%Y%m%d\n", localtime($t));
  $a=localtime($t);
  `echo "$a" >$f`;
}
```

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.

Exercise 5: What outputs will be the result of typing in the following shell command lines (in that order)? Explain why.

```
$ a=1
$ a=2 echo $a ; echo $a
$ a=3 ; echo $a ; echo $a
$ ( a=4 ; echo $a ) ; echo $a
$ { a=5 ; echo $a ; } ; echo $a
$ a=6 bash -c 'echo $a'
$ a=7 ; bash -c 'echo $a'
$ bash -c "echo $a"
$ export a
$ bash -c 'echo $a'
```

Exercise 6: Configure your MCS 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.

Exercise 7: Explain what happens if the command "`rm *`" is executed in a subdirectory that contains a file named "`-i`".

Exercise 8: 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`".

Exercise 9: 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.

2 Text tools

Exercise 10: 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'
```

3 File tools

4 Revision control

Exercise 11: Generate a Subversion 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.

5 Build tools

Exercise 12: 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 13: 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.

6 Perl

Exercise 14: 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.