CHALMERS
UNIVERSITY OF TECHNOLOGY

ASSIGNMENT #5

Introduction to Linux

1. Simple Bash scripts

(Adapted from "Bash Guide for Beginners" by Dr. M. Garrels, chapter 2.)

Write a Bash script that displays the following information: the path to your home directory (use `pwd` for this), the path of the shell that you are using (use an environment variable for this), the total amount of physical memory and swap space, and the number of processes running on the system.

Provide clear information in the output of the script about what is being displayed. Also add comments in the script documenting your code. The instructors will happily look at your final code and give you suggestions.

Apply appropriate permissions to your script and run it. Run the script in normal mode and in debug mode. It should run without errors.

In case your script ran flawlessly from the first time (you are a guru!), introduce a few errors and see what happens if you misspell commands, if you leave out the shebang, if you replace the shebang with an non-existent interpreter, if you misspell shell variable names, or if you write them in lower case characters after they have been declared in capitals. Check what the debug comments say about this.

2. Scripts with options

Now make a copy of the script in Exercise 1 and rework the script so that it works with five optional arguments:

```
scriptname [-h] [-s] [-m] [-p] [--help]
```

The information about home directory, shell, memory, and/or number of processes should only be displayed if the respective options are supplied. If --help is supplied, then a short manual should be displayed (and the other options must be ignored); check `touch --help` for inspiration. The script should return exit code 0 if at least one of these types of information is displayed, exit code 1 if no options were supplied, and exit code 2 if invalid options were supplied.

3. Expansion

(Adapted from "Bash Guide for Beginners" by Dr. M. Garrels, chapter 3.)

1. Create 3 variables, VAR1, VAR2, and VAR3; initialize them with the values "thirteen", "13", and "Happy Birthday", respectively. Export VAR2.

2. Display the values of all three variables.

3. Remove VAR3.

4. Display the three variables in a subprocess with commands like `bash -c "echo \$VAR1"` (why did I put the \ here?). Which variable gets printed on the terminal? Can you see the two remaining variables in a new terminal window? Describe in your own words what `export` does.

5. Edit an appropriate Bash configuration file so you get to read a nice welcome message when you open a terminal and you also get the current date and time automatically.

6. Write a script in which you assign two integer values to two variables. The script should calculate the surface and the circumference of a rectangle of which the sides equal the two variables. Generate elegant output and document your code with comments.

7. Write a Bash script that sums the resident set sizes of all running processes using arithmetic expansion. (You could actually do this with a single `awk` command.)

4. Conditional statements

(Adapted from "Bash Guide for Beginners" by Dr. M. Garrels, chapter 7.)

1. Use an if/then/elif/else construct to print information about a certain month and year (two arguments of the script). The script should print the number of days in this month, and correctly treat the month of February depending on whether or not the year is a leap year. It should also correctly treat September 1752.

2. Do the same as above using case statements.

3. Edit the leaptest.sh script from Section 7.2.4 of the "Bash Guide for Beginners" so that it takes one argument, the year. Test that exactly one argument is supplied. Return an error code if there are less or more arguments, or if the argument is not a year.

4. Write a script called *`whichdaemon.sh`* that checks if the `httpd` and `systemd` daemons are running on your system. If an `httpd` is running, the script should print a message like "This machine is running a web server."

5. Write a Bash script *`samefile.sh`* that takes two file names as arguments and tells you whether the two file names are hardlinks to the same inode.