

# Workplace Technology & Skills (cs3306)

## Assignment 1

Implementing a Script (Due: January 21. Marks: 10)

### Introduction

This assignment is about implementing a filter using *basic* shell functions. When you're finished with this assignment you should know how to: use shell functions which carry out certain well-defined tasks, exit with a proper exit status if the script isn't run properly, remove the script's intermediate file(s) with `trap`, parse the command line argument with `shift`, create temporary files with `mktemp`, ...

### 1 Assignment Details

For this assignment you will implement a script called `grepper`, which is a filter and which copies lines from `stdin` to `stdout` if they match words that are provided as command line arguments. To practise command line parsing, the command line arguments must be a non-empty sequence of elements of the following form:

- `-word [a-zA-Z]+`

i.e. each element should start with `-word`, followed by a sequence of letters.

The *words* of the script are the sequences of letters after the `-word` keys in the command line arguments. The script should take its input line by line and only output the lines that match a simple `grep` on one of the words in the script's words. If the current input line matches several words, the line should be output only once.

The script should buffer its output and should only copy its output to `stdout` when it has processed all user input.

The following shows two examples. The here document in the second example is used for clarity but isn't needed, really.

```
$ ./grepper -word 123
Usage: ./grepper [ -word word ]+
$ echo $?
1
$ ./grepper -word hello -word world <<EOF
> line one hello world
```

```

> line two hello
> line three worlds
> line four moon
> EOF
line one hello world
line two hello
line three worlds
$ echo $?
0
$

```

- **The grepping should be implemented in a while loop which repeatedly greps for the next word in the script's words, until the current line matches a word, or until the loop has exhausted all words.** You will need to do this for every line in the input, so there will be two nested while statements.
- The script should use unix commands only and should not use interpreted or compiled languages such as python, Java, C, ...
- The script should not use user-installed packages.

Needless to say, the script should:

- Not leave behind temporary files;
- Output a proper error message if it's not used properly;
- Exit with a suitable exit status;
- Have a clear structure;
- ...

## Submission Details

- You should know what to put on the first line of your script.
- Lines 3–5 of your script should have a comment like the following:

```

# Name: Fill in your name.
# Number: Fill in your student ID.
# Assignment: Assignment Number.

```

- Use the cs3306 moodle site to upload your program as a single *.tgz* archive called *Lab-1.tgz* before 23:55pm, January 21, 2019. To create the *.tgz* archive, do the following:
  - ★ Create a directory *Lab-1* in your working directory.
  - ★ Copy grepper into the directory. Do not copy any other files into the directory.
  - ★ Run the command `'tar cvfz Lab-1.tgz Lab-1'` from your working directory. The option 'v' makes tar very chatty: it should tell you exactly what is going into the *.tgz* archive. Make sure you check the tar output before submitting your archive.
  - ★ Notice that file names in Unix are case sensitive and should not contain spaces.
- Notice that the format is *.tgz*: do *not* submit zip files, do *not* submit tar files, do *not* submit bzip files, and do *not* submit rar files. If you do, it may not be possible to unzip your assignment.
- Marks are deducted for poor choice of variable names and/or poor layout.

- No marks shall be awarded for scripts which are rejected because of syntax errors.