



MODULAR PROGRAMME
ASSESSED COURSE-WORK SPECIFICATION

Module Details:

Module Code: UQC109S2	Module Title: Computer Networks & Operating Systems	
Module Leader: Nigel Gunton		
Module Tutors: Ian Johnson John Counsell Nigel Gunton Fred Stinchcombe Graham Darke		
Assignment CW1	Element Number: Weighting 25%	Total Assignment Time: 12 hrs + lab time

Dates:

Date assignment issued to students: Oct 21st	Date for return of marked work: w/b 03/02/03
Submission Place: post-box in N foyer, below the North stairs	Date of Submission: w/b 16/12/02
	Time of Submission: 10.00am

Deliverables:

As listed on the Assignment spec sheet

Overview:

Even in these days of Graphical User Interfaces (GUIs) most modern operating systems still offer a command line interpreter, or shell. Many systems administrators are frequent users of command line interfaces, even on NT! From a learning point of view, command-line interfaces provide an opportunity to study the underlying operating system calls and to this end you will be developing a simple command line interpreter, or shell. Your shell should provide a prompt, error/usage messages for any built-in commands and pass other commands to the underlying system for execution. By default, output from the commands should be to stdout (the terminal screen).

- This assignment will be developed in stages and signed off by your lab tutor as you proceed.

**THERE WILL BE NO SIGN-OFFS FOR SECTION ONE AFTER FRIDAY 6th DECEMBER.
NO SIGN-OFF == NO MARK FOR THAT SECTION.**

- You will be expected to work on this during your lab sessions **AND** in your own time.

Requirements:

You can tackle either section 1a or section 1b first.

Section 1a)

Use the code provided at

<http://www.hawklord.uklinux.net/system/shell/cs3.htm>

- i) paste it into a single program.
- ii) Add your own comments.
- iii) Compile and debug it.
- iv) demonstrate the compiled code and your understanding of how it works to the lab tutors satisfaction.

Completing this successfully will accrue 20%.

Alternatively you may write your own shell from scratch to provide the same basic functionality as the provided code.

Section 1b)

To develop the following elements, initially as stand-alone programs, in C, but eventually to be built into your shell. It will pay to think ahead and to consider functions that will be common to all/many of the stand-alone versions.

`pwd` This should print, on stdout, the path to the current directory.

`cd` This should take an optional path as an argument. If no argument is provided then the default behaviour is to change directory to the users home directory.

`ls` The 'list directory contents' command. It should accept the flags `-a` and `-l` and respond appropriately. (see worksheet 3 for some clues).

`ps` Default behaviour is to list all processes owned by the user. It should accept the flag `-A` as an argument and list all current processes and their process Id's .

`kill`

This command should respond as follows :

```
kill pid
    send SIGTERM to process pid
```

```
kill -l
    list the signals sent by this command. Your version of kill should recognise
    SIGTERM, SIGKILL and SIGHUP. It should provide a list of both the names of
    the signals and their numbers. (man 7 signal†)
```

```
kill signal pid
    Send the specified signal to pid. It should recognise both the numeric value and
    the name of the signals.
```

[†] man -s 3HEAD signal on Solaris

These should be demonstrated separately and explained to your lab tutor who will sign and date the relevant section of the sign-off sheet when s/he is satisfied with your code. This will accrue a maximum of 50% (10% per command).

The marks breakdown for this part is as follows :

For each component :

Comments
up to 2 marks

Structure
up to 3 marks

Correct functionality
up to 5 marks

Integration of the above into a simple command line interpreter, or 'shell' using either your own code or the code from 1a). This should use exec calls to execute any non built-in commands

up to 10 marks

Section 2)

EITHER

- a) The extension of your shell program to handle redirection of stdin, stdout & stderr, pipes and background execution of commands (10%).

OR

- b) The implementation of a remote shell based on your shell. This requires the development of a shell daemon which will respond to requests from clients and then execute your shell, the development of the client program. The extension of your server and client to allow the server to spawn a process and then to continue listening for further connection requests (10%).

A further 10% will be assigned at the lab tutors discretion for additional work, own shell, or other factors.

Constraints :

- 1) All code **MUST** be demonstrated and explained to your lab tutor before it will be signed off.
- 2) Remember, this is an individual assignment and that assessment offences are taken seriously. This does not prevent you from discussing problems and ideas with your peers and you are encouraged to do so as long as the final result is your own work. If you use sections of code from other sources then they must be clearly identified, contain your own comments and you will be expected to demonstrate your understanding of the code to your lab tutor.

Support :

C programming

The syntax of C and the syntax of Java are very similar. In addition there is an excellent C tutorial on Ian Johnsons home-page and many other web based tutorials. In general, picking up another programming language is something you should be confident about as students on a Computing degree. Most languages share the same concepts and control structures so you should only need to focus on the syntax.

If you wish you can use the integrated development environment 'xwpe' that will provide editing, compiling and debugging. It is similar to the Borland Turbo C environment. However it is recommended that you use emacs and gvd.

system calls

These will be covered in the lectures. There is also a very good web-site that covers much of the assignment material.(see link from my home page).

your lab tutor

Will provide support with syntax problems etc.

think!

For each of the first sections you need to think about

- what information do you need to extract from the system?
- where is that information to be found?
- what order do you need to do things in?

For example with the change directory command 'cd' on its own, it will return the current user to their home directory. So you need to find out who the current user is and where they live (their home directory). This information can be obtained from the users environment through the use of the system call `getenv()`. This takes the name of an environment variable (see `man environ`) as an argument and returns a pointer to the value in the environment (usually a string).

- `man 2 intro` will give an overview of the system calls.
- `man syscalls` will give a list of all 164 system calls
- `man 3 intro` will give an overview of the C library calls.
- `man stdio` covers the calls in `stdio.h`.

You can then look at the individual man page for a call to obtain even more detail.

- `man proc` could come in handy.

Deliverables :

- a) Your sign-off sheet, signed and dated for all completed work.
- b) Copies of all code that has been demonstrated/explained to your tutor. This code **MUST** be signed by your tutor.

UQC109S2 Computer Networks & O/S Course-work 2 2002 - 2003

Course-work Checklist 1b)

Student Number

	Total
ls Comments : Structure : Function :	<input type="text"/>
cd Comments : Structure : Function :	<input type="text"/>
ps Comments : Structure : Function :	<input type="text"/>
pwd Comments : Structure : Function :	<input type="text"/>
kill Comments : Structure : Function :	<input type="text"/>
Demonstration of integration of above :	<input type="text"/>
Demonstration of extended design :	<input type="text"/>

Lab Tutor Signature.....

UQC109S2 Computer Networks & O/S Course-work 2 2002 - 2003

Course-work Checklist 1a)

Student Number

	Total
Standard of Comments :	<input type="text"/>
Execution :	<input type="text"/>
Runs :	<input type="text"/>
Structure :	<input type="text"/>
Function :	<input type="text"/>
Understanding :	<input type="text"/>

Lab Tutor Signature.....