

R228/417

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8.

BSc Computer Science

Year 4

SUPPLEMENTAL EXAMINATIONS 2005

SYSTEMS PROGRAMMING

Mr. C. O'Leary

Dr. B. O'Shea

Mr. B. Kearney

September 2005, 2:30 – 5:30

Answer **FOUR** questions

All questions carry equal marks

1. (a) Describe the set of processes that start when a user logs in to a UNIX system and the relationship between these processes. **(7 marks)**
 - (b) Perform a comparison of the Bourne shell and the C shell scripting languages. **(8 marks)**
 - (c) Write a simple Bourne shell script that will take a directory name as a command line argument, and print out the total size of all the ordinary files in that directory. **(10 marks)**
2. (a) Provide a detailed discussion of the UNIX terminal. **(9 marks)**
 - (b) Consider the code shown below:

```

#include <sys/types.h>
#define SMALLSZ 10

int main(int argc, char **argv) {
    ssize_t nread;
    char smallbuf [SMALLSZ + 1];

    while((nread = read(0, smallbuf, SMALLSZ)) > 0) {
        smallbuf[nread] = '\0';
        printf("nread:%d %s\n", nread, smallbuf);
    }
}

```

The above code will read from the terminal in canonical mode. Change the code so that it reads from the terminal in raw mode, and explain the significance of this change.

(8 marks)

- (c) With the terminal set to raw mode, and the user entering the following string:

The quick brown fox

what output would you expect from your program in part (b) above if the MIN and TIME parameters were set as follows:

	MIN	TIME
(i)	0	0
(ii)	8	0
(iii)	0	8
(iv)	8	8

(8 marks)

3. (a) A file `f` has permissions `06600` associated with it. Explain in detail the significance of each bit of those permissions.

(7 marks)

- (b) In relation to file management, describe each of the following, using sample code where appropriate:

- (i) File descriptor
- (ii) File creation mask
- (iii) Hard link
- (iv) Symbolic link
- (v) `stat` structure
- (vi) inode structure
- (vii) Block device file
- (viii) Character device file

(8 marks)

- (c) Consider the following code:

```
#include <stdlib.h>
#include <sys/stat.h>
#include <dirent.h>

int main(int argc, char **argv) {
    int ftw2(const char *path, int (*func)());
    int list(const char *name, const struct stat *sptr, int type);
    ftw2(argv[1], list);
}

int ftw2(const char *path, int (*func)()) {
    struct stat statbuf;
    stat(path, &statbuf);
    // call the function whose name is passed as an argument
    (*func)(path, &statbuf, 1);
}

// type = 0 for directory, type = 1 for file
int list(const char *name, const struct stat *status, int type) {
    if(type == 1)
        printf("%-40s\t0%-3o\n", name, status->st_mode & 0777);
    else
        printf("%-40s*\t0%-3o\n", name, status->st_mode & 0777);
}
```

The above code contains a function named `ftw2(...)`, which is an attempt at re-implementing the `ftw(...)` function available in the C library `ftw.h`.

- (i) Describe the behaviour of the above program.
- (ii) Rewrite the `ftw2(...)` function so that it has the same behaviour as the `ftw(...)` function.

(10 marks)

4. (a) Describe in detail the theoretical operation of semaphores. You do not need to include any code in your answer. **(6 marks)**
- (b) Using sample C code, show how semaphores can be used on UNIX systems to enforce mutual exclusion of protected regions between processes. **(9 marks)**
- (c) Show how semaphores can be used to provide the following functionality:
- Three processes start, each performing some operations.
 - When each process finishes its work it blocks *without* exiting.
 - A fourth process executes after a while telling each blocked process to exit.
- (10 marks)**
5. (a) Using examples and sample code, describe in detail how a UNIX process creates child processes, synchronises with child processes by waiting and also how a process changes its program code. **(6 marks)**
- (b) Describe, compare and contrast the following three mechanisms for exchanging data between two processes on the same host
- (i) Pipes
 - (ii) FIFOs / named pipes
 - (iii) Message queues
- You must use diagrams, examples and sample code in your answer. **(9 marks)**
- (c) Show how it is possible to implement *connection-oriented* and *connectionless* communication between processes on separate hosts.
- Describe the circumstances under which connection-oriented or connectionless communication would be appropriate. **(10 marks)**

6. (a) Consider the following session:

```
[examiner@host q3]$ ls -l
total 12
-rwx----- 1 examiner grp      80 Jan  1 10:00 q3a_1
-rw----- 1 examiner grp    234 Jan  1 10:00 q3a_2.c
-rwx----- 1 examiner grp     27 Jan  1 10:00 store
[examiner@host q3]$ cat q3a_1
#!/bin/sh

trap 'store' 16

while [ 1 ]
do
    sleep 1
done
[examiner@host q3]$ cat store
#!/bin/bash

date >> log

[examiner@host q3]$ cat q3a_2.c
#include <stdio.h>
#include <signal.h>
int main(int argc, char** argv) {
    char input[10];
    int pid = atoi(argv[1]);
    do {
        printf("Enter info > ");
        fgets(input, 10, stdin);
        kill(pid, 16);
    } while (input[0] != 'q');
}
[examiner@host q3]$ cc -o q3a_2 q3a_2.c
[examiner@host q3]$ q3a_1 &
[1] 12624
[examiner@host q3]$ q3a_2 12624
Enter info > line1
Enter info > line2
Enter info > line3
Enter info > quit
[examiner@host q3]$
```

Describe in detail what happens during the session shown above.

(7 marks)

(b) In the context of signals and signal processing explain what is meant by the following terms:

- (i) Signal handling
- (ii) Signal blocking
- (iii) Normal termination
- (iv) Abnormal termination
- (v) Core dump

(8 marks)

- (c) Consider the following system:
- (i) A server process monitors the status of a file by checking every second whether its size has changed.
 - (ii) Client processes (NOT children of the server) are interested in finding out when the size of the file has changed, but don't care what the actual size of the file is.
 - (iii) When client processes start up they must inform the server process that they exist by getting their process ID to the server.
 - (iv) The whole system must be implemented using signals as the only inter-process communication mechanism.

Some code for the server is shown below. Show how to fully implement the server and provide the code for the client.

```
#include <stdio.h>
#include <sys/stat.h>
#include <signal.h>

int client_pids[100];
int num_clients = 0;

int main(int argc, char **argv) {

    void inform_children();
    int file_size;
    char *filename;

    filename = argv[1];
    struct stat statbuf;
    stat(filename, &statbuf);
    file_size = statbuf.st_size;

    while(1) {
        if(statbuf.st_size != file_size) {
            inform_children();
            file_size = statbuf.st_size;
        }
        sleep(1);
    }
}

void inform_children() {
```

(10 marks)