

S228/417P

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8.

BSc Computer Science

Year 4

SUMMER EXAMINATIONS 2004

SYSTEMS PROGRAMMING

Mr. C. O'Leary
Dr. B. O'Shea
Mr. D. Yeates

26th May 2004, 2:30 – 5:30

Answer **FOUR** questions

All questions carry equal marks

1. The code shown below will be used for *Part (a)* of **Question 1**. Assume the existence of a file named `data_1`, which is of size 1000 bytes exactly.

```
#include <unistd.h>
#include <fcntl.h>

main () {
    int fd_1, fd_2;
    pid_t pid;
    char buf[100];
    char line[5];

    fd_1 = 0;
    fd_2 = 1;

    fd_1 = open("data_1", O_RDONLY);
    printf("1: %d %d\n", fd_1,
           lseek(fd_1, (off_t)0, SEEK_CUR));

    read (fd_1, buf, 100);
    printf("2: %d\n", lseek(fd_1, (off_t)0, SEEK_CUR));

    if(!fork()) {
        sleep(5);
        printf("3: %d\n",
               lseek(fd_1, (off_t)0, SEEK_CUR));
        lseek(fd_1, (off_t)-100, SEEK_END);
        read(fd_1, buf, 10);
        printf("4: %d\n",
               lseek(fd_1, (off_t)0, SEEK_CUR));
        // sprintf will write into 'line' char string
        sprintf(line, "5: %d\n", fd_2);
        write(fd_2, line, 5);
    }
    else {
        fd_2 = creat("data_2", 0644);
        printf("6: %d\n",
               lseek(fd_1, (off_t)500, SEEK_SET));
        printf("7: %d\n",
               lseek(fd_2, (off_t)5, SEEK_SET));
        sprintf(line, "8: %d\n", fd_2);
        write(fd_2, line, 5);
    }
}
```

- (a) What output would you expect to be generated by the above program, following successful compilation and execution? Explain your answer in full.

(11 marks)

- (b) Describe *in full* how both `wait(...)` and `waitpid(...)` can be used to both synchronize processes, and pass certain information between them. You must use sample code in your answer.

(8 marks)

- (c) UNIX uses a scheduling algorithm to decide which process should get control over the CPU at any point in time. Give a brief overview of this algorithm, and using sample code, explain how you can assist the scheduler when you write your programs. **(6 marks)**
- 2 (a) Using diagrams and code, explain how a UNIX shell program uses pipes to redirect standard output from one process into the standard input of another process. **(10 marks)**
- (b) You are required to create a server process that will accept connections from unrelated client processes in order to perform some operation. Describe in detail *three* different strategies that could be used to develop both client and server. Discuss the strengths and limitations of each approach. **(15 marks)**
3. (a) Using sample code show how a process can send a signal to
- i. itself,
 - ii. another process.
- Under what circumstances will these calls fail? **(5 marks)**
- (b) For most signals sent by the Kernel, the default action for the receiving process is termination. Distinguish between normal and abnormal termination, and using code show how it is possible for a process to
- i. ignore a signal,
 - ii. block a signal temporarily,
 - iii. handle a signal. **(10 marks)**
- (c) Why are *session groups* of such importance for interrupt signals? In your answer, you should outline how the members of a session group are determined, and their relationship to the terminal. **(10 marks)**

4. The code shown below will be used for *Part (a)* of **Question 4**. Assume the existence of a file named `data_4`, which is of size 100 bytes exactly.

```
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

main() {
    int fd;
    struct flock first_lock;
    struct flock second_lock;

    first_lock.l_type = F_WRLCK;
    first_lock.l_whence = SEEK_SET;
    first_lock.l_start = 40;
    first_lock.l_len = 50;

    second_lock.l_type = F_WRLCK;
    second_lock.l_whence = SEEK_SET;
    second_lock.l_start = 39;
    second_lock.l_len = 1;

    fd = open("data_4", O_RDWR);

    fcntl(fd, F_SETLKW, &first_lock);

    if(fork()) {
        sleep(2);
        fcntl(fd, F_SETLKW, &second_lock);
    }
    else {
        fcntl(fd, F_SETLKW, &second_lock);
        fcntl(fd, F_SETLKW, &first_lock);
        exit(0);
    }
}
```

- (a) Using diagrams, explain exactly what will happen when the above code is executed.

(10 marks)

- (b) Explain what is meant by a *key* in reference to advanced inter process communication facilities.

(4 marks)

- (c) Discuss how, in theory, the $p()$ and $v()$ operations on a *semaphore* can be used to protect a critical region of code.

(6 marks)

- (d) Using sample code, demonstrate how semaphores can be implemented using C on UNIX.

(5 marks)

5. (a) Distinguish between the *real user ID*, *real group ID*, *effective user ID* and *effective group ID* of a UNIX file. Discuss how permissions can be set to allow a process to assume an effective ID when executed. **(6 marks)**

(b) Using `stat` and `ftw`, write a C program that will traverse a directory structure and print out the number of files in that directory tree, as well as the total size of all the files. Explain all functions and arguments used. **(7 marks)**

(c) Identify and describe in full the various components of a UNIX file system. **(7 marks)**

(d) Using sample code and examples, demonstrate the differences between *hard links* and *symbolic links*, in terms of UNIX files. **(5 marks)**

6. (a) Describe what takes place between the time a UNIX machine is turned on and when the user can start using the shell. **(9 marks)**

(b) Using examples and sample code, demonstrate what is meant by the following, in terms of programming with the Bourne shell:

- Command substitution
- Exporting environment
- The Here Document

(6 marks)

(c) Write a simple Bourne shell script that will take a directory name as a command line argument. The total size of all ordinary files in the directory should be calculated, as well as a count of the number of directories. Both pieces of information should be printed out.

(You can assume the `ls -l` command will display file information in the following format:

```
-rw-r--r--    1 myname  mygrp   1000 May 26 12:00 filename
```

(10 marks)