

R228/417

DUBLIN INSTITUTE OF TECHNOLOGY  
KEVIN STREET, DUBLIN 8.

---

# BSc Computer Science

Year 4

---

SUPPLEMENTAL EXAMINATIONS 2004

---

SYSTEMS PROGRAMMING

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

Date/Time to be provided – 3 hours

Answer **FOUR** questions

**All questions carry equal marks**

1. (a) How does UNIX decide which process should execute at any point in time?  
(4 marks)
- (b) How can message queues be used by processes to exchange data? Use sample C code in your answer. What advantages / disadvantages do they offer when compared with FIFOs?  
(10 marks)
- (c) Show how *sockets* can be used by processes on separate machines in order to communicate. Discuss the different types of socket, and use sample C code in your answer.  
(11 marks)
2. (a) Describe in full *three* mechanisms for synchronizing a group of processes. You must use an example and sample C code in your implementation.  
(13 marks)
- (b) Show how *record locking* can be used to prevent processes from simultaneously accessing the same part of a given file. Why is it important that such a mechanism exist? What are the potential problems that exist when using this mechanism, and to what degree are they handled by UNIX?  
You must use diagrams and sample C code in your answer.  
(12 marks)
3. (a) What processes are started in what order when a user logs in to a UNIX machine?  
(7 marks)
- (b) When a signal is generated by a keyboard interrupt, to whom is this signal sent? How does this impact on the various processes that are currently running?  
(7 marks)
- (c) In terms of signals discuss the following:  
i. Different types of signals  
ii. Consequences of receiving a signal  
iii. Handling signals  
iv. Ignoring signals  
v. Blocking signals  
You should use sample C code in your answer  
(11 marks)

4. (a) Explain fully what will happen when the code below is executed. Discuss the functions and arguments used.

```

#include <sys/time.h>
#include <sys/wait.h>
#include <unistd.h>

void a(int [][]);
int b(int []);

main() {
    int pip[5][2];
    int i;
    for(i = 0; i < 5; i++) {
        pipe(pip[i]);
        if(!fork())
            b(pip[i]);
    }
    a(pip);
    exit(0);
}

void a(int p[5][2]) {
    char buf[52], ch;
    fd_set set, master;
    int i;

    for(i = 0; i < 5; i++)
        close(p[i][1]);

    FD_ZERO(&master);
    FD_SET(0, &master);

    for(i = 0; i < 5; i++)
        FD_SET(p[i][0], &master);

    while(set = master, select(p[4][0], &set, NULL, NULL, NULL) > 0) {
        for(i = 0; i < 5; i++)
            if (FD_ISSET (p[i][0], &set))
                if(read(p[i][0], buf, 52) > 0) {
                    buf[51] = '\n';
                    printf("Message from child%d\n", i);
                    printf("\t%s\n", buf);
                }

        if(waitpid(-1, NULL, WNOHANG) == -1) return;
    }
}

int b(int p[2]) {
    int count;
    close(p[0]);
    char message1[52];
    char message2[52];
    int sleep_for = (getpid() % 10);
    sprintf(message1, "My pid is %d, yours is %d, I'll exit in %d secs\n",
        getpid(), getppid(), sleep_for);
    write(p[1], message1, 52);
    sleep(sleep_for);
    sprintf(message2, "I have now exited, my pid is %d, good bye for now\n",
        getpid());
    write(p[1], message2, 52);
    exit(0);
}

```

**(10 marks)**

- (b) Discuss fully the exec system call.

**(5 marks)**

- (c) Using sample code and diagrams, show how it would be possible to implement the command piping mechanism supported by UNIX shells.

**(10 marks)**

5. The code shown below will be used for *Part (a)* of **Question 5**. Assume the existence of a file named `data_1`, which is of size 2000 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) 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.

**(8 marks)**

- (c) Using sample code and examples, demonstrate the differences between hard links and symbolic links, in terms of UNIX files.

**(6 marks)**

6. (a) Discuss UNIX file permissions. You should explain how the permissions associated with a file impact on the file when it represents:
- i. An ordinary file
  - ii. A directory
  - iii. A program

Your discussion should include references to *effective* and *real* IDs.

**(8 marks)**

- (b) Discuss in full what is meant by *shell scripting*. Why is this a useful facility in UNIX environments?

**(7 marks)**

- (c) Write a shell script in any shell language that will count the number of arguments passed in at the command line, and then search the current directory and all immediate subdirectories for files that are named in the list. The size of each of these files should be printed out.

**(10 marks)**