PAPER CODE NO. COMP204	EXAMINER : **DR D Jackson** DEPARTMENT : Computer Science Tel. No. **3678**

SUMMER 2000 EXAMINATIONS

Bachelor of Science : Year 2

SYSTEMS PROGRAMMING

TIME ALLOWED : TWO Hours

INSTRUCTIONS TO CANDIDATES

Answer *eight* questions from Section A and *two* questions from Section B.

If you attempt to answer more than the required number of questions (in any section), the marks awarded for the excess questions will be discarded (starting with your lowest mark).

SECTION A

Give brief and relevant answers to any **eight** of the following ten questions:

[5 marks each]

(1) What C functions are available for dynamic allocation and freeing of memory, and how are these routines used?

(2) How does the *make* utility decide whether work needs to be done to bring a software application up to date?

(3) How could you check whether a call to *exec()* had failed?

(4) Why can't hard links be used across filesystems, and what is the remedy?

(5) What is meant by *mutual exclusion*, and when might it be needed?

(6) What information does UNIX hold in its */etc/passwd* file?

(7) Give two reasons why error reporting should be done via the *stderr* stream rather than the *stdout* stream.

(8) What purpose do the SUID and SGID bits serve in the file mode word?

(9) How is an orphan process created, and what is its usual fate?

(10) Give three examples of faults that might cause a process signal to be generated on a UNIX system.

SECTION B

This section calls for fuller solutions than those of the previous section.

Answer any **two** of the following three questions:

[30 marks each]

B1

(a) A user sits down at a character-based UNIX terminal displaying a 'login' prompt, and proceeds to log on. Describe the process activity that takes place under UNIX from the time the user sits down to the time he or she is issued with a command prompt.

[7 marks]

(b) The user then runs the following sequence of commands. Describe the actions performed by the shell and UNIX in executing each one. [4 marks each]

```
(i)   cat *.out > text1
(ii)  rshow | wc -l
(iii) netscape &
(iv)  for i in *.c; do cp $i $i.bak; done
```

(c) The user now logs out. Again, describe the process activity that takes place when this occurs. [5 marks]

(d) Consider the command in (b)(ii) above (i.e. *rshow | wc -l*). What would be the simplest way of executing this command from within a C program? [2 marks]

B2

(a) Under UNIX, files can be accessed via file *descriptors* or file *streams*. Describe the differences between the two approaches, and give the advantages and disadvantages of each. [5 marks]

(b) File descriptors are also used as a consistent interface for other forms of input and output. For each of the following, detail the steps involved in setting up communication, focusing in particular on the role that file descriptors play:

(i) Pipes [7 marks]
(ii) Sockets [7 marks]
(iii) Peripheral devices [4 marks]

(c) It is also possible to re-direct a file descriptor from one source or destination to another. Explain how this can be achieved from within a C program. [7 marks]

B3

Eddie Clueless, a novice programmer, is having difficulty in writing his C program, and has come to you for some expert assistance. He has been asked to write a program (called 'bigmac') that will act as a simple macro processor. By writing C or pseudo code fragments, explain to Eddie how he can solve each of the following sub-tasks.

(a) Eddie's first problem is to process the command line. Typically, this will look like:

```
$ bigmac infile
```

Show Eddie how he can access the contents of the command line, open the given file, and do all the appropriate error checking. [7 marks]

(b) The main job of the program is simply to output the contents of the file until it detects either a macro definition or macro replacement. A macro definition looks like this:

```
@NAME=Eddie Clueless
```

After this has been processed, any occurrence of "%NAME" in the file should be replaced by "Eddie Clueless".

Explain to Eddie how to write this part of the program, including details of the data structures that will be used to store the macro definitions. [14 marks]

(c) An extension to the above is to replace a UNIX command by the output of that command. Hence, the string "%(ls –l)" in the file should be replaced in the program's output by the actual directory listing. Again, show Eddie how this can be done, making sure that your approach is one that emphasises efficiency. [9 marks]