



Academic Year: 2007 - 2008
Examination Period: January
Module Leader: Ian Johnson
New Module No: UFE-EHJ-30-2
Old Module No: UFS-EHJ-30-2
Title of Module: Operating Systems & System Administration

Examination Date:
Examination Start time: 09:00
Duration of Examination: 2 Hour(s) 00 Minutes

Instructions to Students: Attempt all of Section One and any 1 question in Section Two

Materials supplied to the student will be:

Number of Examination Booklets (+ any continuation booklets as required) per Examination	1
Number of Pre-printed OMR (Multiple Choice Answer Sheet)	0
Number of sheets of Graph Paper size G3 (Normal)	0

Additional Instructions to Invigilators:

Calculators may be used subject to University regulations	Yes
Students allowed to keep Examination Question Paper	No
Material supplied by student allowed :	None
Additional Specialised Material:	

Treasury tags & adhesive triangles will be supplied as standard

This page blank

SECTION ONE

Attempt all questions in this section.

- 1)** Briefly explain the role of the 'shell' or command-line interpreter and the effect that the special characters '&' and '|' have on it. (5 Marks)

- 2)** Briefly explain which key Unix commands are built into the shell and why this is. (5 Marks)

- 3)** The use of system calls requires a different approach to error handling. Describe the process by which the kernel provides such error information and the calls that enable the programmer to access this information. (5 Marks)

- 4)** Scheduling policies can be Non-Preemptive, Preemptive or Cooperative. Briefly explain each of these policies. (5 Marks)

- 5)** A programs access to a device can be part of the user process as in GNU/Linux or via a separate process as in Microsoft's XP. What effect do the different approaches have on on task switching. (5 Marks)

SECTION TWO

Answer any one question in this section.

6) Unix implementations provide a mixture of approaches to InterProcess Communication (IPC) as well as mutual exclusion through the use of semaphores.

i) Describe the following forms of IPC. You should indicate whether the communication is limited to related processes only, and also the extent to which mutual exclusion is provided by the kernel.

- `int pipe(int filedes[2]);`
- `mkfifo [options] file...`
- messages :
 - `int msgsnd();`
 - `ssize_t msgrcv();`
- shared memory:
 - `int shmctl(); int shmget(); etc.`
 - `void * mmap();`

(15 Marks)

ii) A semaphore is used to control access to a shared data object. Describe how semaphores function, the steps a process must take both to create a semaphore and its use to gain access to a shared resource. You may make reference to POSIX compliant semaphore calls.

(10 Marks)

7)

- i) Pages sometimes need to be removed from memory in order to make space for another process's pages. There are a number of different algorithms that could be used, these include the FIFO, modified FIFO and NRU algorithms. Describe how these three work, along with any advantages or disadvantages. (9 Marks)
- ii) Memory Management is an important component of modern operating systems, enabling multiple processes to share physical memory. GNU/Linux makes use of both a segmented memory model and a paged memory model. Provide labelled diagrams to show how a C program is mapped into virtual memory segments and how these relate to the paged memory. (6 Marks)
- iii) The architecture dependent part of GNU/Linux on the i386 architecture uses paged memory management. Explain, with the aid of diagram (i), how the page table structure is organized. Your answer should include an explanation of the fields in the page table entry. (10 Marks)

(10 Marks)

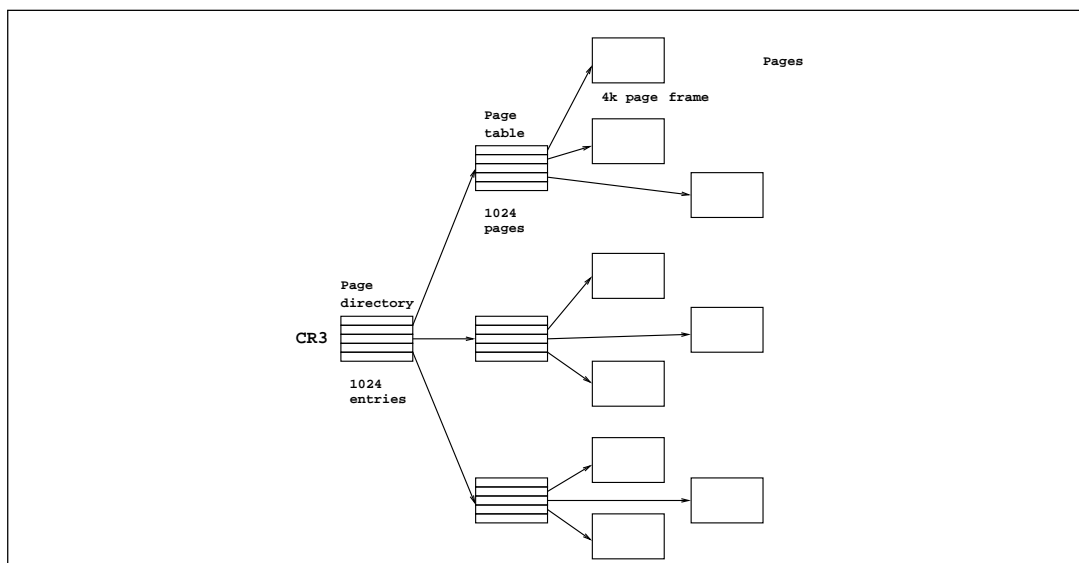


diagram (i)

8)

- i) Describe, with the aid of a clearly labelled diagram, the way in which a C program is started by the kernel, the ways that the program can exit or terminate and the way in which exit handlers can be used.
(12 Marks)
- ii) Explain the behaviour of the parent and child processes in the following code fragment. You should make clear the role played by the `exec` family of calls.
(13 Marks)

```
if ((pid = fork()) < 0) {
    perror("fork");
    exit(1);
}
if (pid == 0) {
    execvp(*args, args);
    perror(*args);
    exit(1);
}
waitpid(pid,&status,0) != pid;
if WIFEXITED(status)
    printf("return value was %d \n",WEXITSTATUS(status));
```