Name ______________________ Student No. ______________________

No aids allowed. Answer all questions on test paper. Use backs of sheets for scratch work.

Total Marks: 70

[20] 1. **Processes:**

(a) What is a context switch? Describe the actions taken by the kernel to perform a context switch between processes.

(b) Consider a computer system which has multiple register sets. Describe the actions of a context switch if the new context is already loaded into one of the register sets.

(c) In the above case (i.e., multiple registers), what else must happen if the new context is in memory rather than in a register set, and all the register sets are in use?

(d) Describe the different IPCs (Interprocess Communication) mechanisms in Linux.

(a) A context switch is the act of switching the CPU from one process to another. It requires saving the state of the old process (i.e, its PCB) and loading the PCB of the new process.

(b) The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time.

(c) In that case, the context switch is just as with only one set of registers: one set of registers must be chosen, its contents saved, and the saved process loaded into these registers.

(d)
- Pipes and FIFOs: Best suited to implement producer-consumer interactions among processes. Some processes fill the pipe with data while other extract data from the pipe.
- Semaphores: the User Mode version of the kernel semaphores.
- Messages: Allow processes to exchange messages (short blocks of data) in an asynchronous way. They can be thought of as signals carrying additional information.
- Shared memory regions: Best suited to implement interaction schemes in which processes must share large amounts of data in an efficient way.
- Sockets: They were introduced initially to allow data communication between application programs and the network interface. They can also be used as a communication tool for processes located on the same host computer; the X Window System graphic interface, for instance, uses a socket to allow client programs to exchange data with the X server.

[20] 2. **Threads:**

(a) Provide two programming examples of multithreading that improve performance over a single-threaded solution.

(b) Provide two programming examples of multithreading that do *not* improve performance over a single-threaded solution.

(c) Suppose that `counter` is a global variable in a multi-threaded program. Discuss the possible problems, and how to fix them (in the context of Pthreads).

(d) What is the difference between kernel and user threads?

(a) A Web server that services each request in a separate thread. A parallelized application such as matrix multiplication where different parts of the matrix may be worked in parallel.

(b) Any kind of sequential program is not a good candidate to be threaded (e.g., a program that computes a tax return). Another example is a "shell" program (such as bash or korn). A shell must closely monitor its own working space such as open files, environment variables, and current working directory.

(c) The problem with a global variable is that several threads may want to access it simultaneously, and so, its value may be undefined. The way to fix this problem with the Pthreads library, is to serialize the access to the variable with the `mutex` function.

(d) User-lever threads are unknown by the kernel, whereas the kernel is aware of kernel threads. User threads are scheduled by the thread library (such as Pthreads for C) and the kernel schedules kernel threads. Kernel threads need not be associated with a process whereas every user thread belongs to a process.

[15] 3. **Process Scheduling:**

(a) Consider the following set of processes, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Priority
P_1	5	8
P_2	3	1
P_3	3	1
P_4	2	1
P_5	7	2

The processes are assumed to have arrived in the order given. Draw a Gantt chart illustrating the execution of these processes using the Round Robin scheduling algorithm with time slice 1.

(b) In the above situation it was assumed that all the process arrived at time 0 in the ready queue, and no new process arrive until they all terminate. In general, new processes arrive continuously; how does this affect the Round Robin algorithm?

(c) Explain the details of the Linux time-sharing process scheduling algorithm. In particular, what is the "recrediting operation," and what is the recrediting formula.

(a)

1	2	3	4	5	1	2	3	4	5	1	2	3	5	1	5	1	5	5	5

(b) RR is designed for the continuous arrival of processes; new processes are put at the tail of the ready queue, and will be scheduled in the next "loop."

(c) For time-shared processes, Linux uses a prioritized, credit-based algorithm. Each process possesses a certain number of scheduling credits.

When a new task must be chosen to run, the process with the most credits is selected. Every time that a time interrupt occurs, the currently running process looses one credit. When the number of credits reaches zero, it is suspended and another process is chosen.

If no runnable processes have any credits, then Linux performs a recrediting operation, adding credits to every process in the system (rather than just to the runnable ones), according to the following rule:

$$\text{credits} = \frac{\text{credits}}{2} + \text{priority}$$

This algorithm mixes two factors:

i. history
ii. priority

One-half of the credits that a process still holds since the previous recrediting operation will be retained after the algorithm has been applied. This way, it retains some history of recent behavior.

Background batch jobs can be given a low priority. Interactive, or I/O bound processes, for which a rapid response time is important, can be given a higher priority.

[15] 4. **UNIX Filesystem:**

(a) The file containing this test has the following properties:

```
228486 -rw-rw-r-- 1 soltys soltys 2461 Mar 13 11:21 test2.tex
```

Explain the meaning of all the fields. In particular, explain the permissions given by `-rw-rw-r--`.

(b) The file `/etc/passwd` contains entries of the form:

```
you:gkmbCTrJ04COM:500:500:You:/home/you:/bin/bash
```

where the second field contains the password information. Explain why this is not a security threat, even though all users have read permission for this file.

(c) The file `/dev/hdc` is the CD-ROM device driver. Explain the concept of mounting it on the file system.

(a) The first field is the inode number of the file. The second field contains the following permissions: the owner (soltys), and the owner's group (soltys) can read and write the file, and everyone else can just read it. The file has one hard link (itself), the owner and group it belongs to are soltys, and soltys, respectively. The size in bytes is 2461, it was created on March 13, at 11:21.

(b) This is not a security threat because this is the encrypted version of the password. When the password is entered, say as x, the value $f(x)$ is computed, where f is a one-way-function, and $f(x)$ is compared with `gkmbCTrJ04COM`; if they are equal, x must have been the correct password. Since f is a one-way-function, it is not easy to compute $f^{-1}(\texttt{gkmbCTrJ04COM}) = x$.

(c) To be able to access the directory structure on a CD, it has to be mounted on the file system, with the command:

```
mount -t iso9660 /mnt/cdrom /dev/cdrom
```

where iso9600 is the filesystem on CD-ROMs, and `/mnt/cdrom` is the directory to which the file structure on the CD has been attached. The CD is now just another directory, as far as the user is concerned.