**MỤC LỤC**

[LAB 1. OS INTRO 3](#_Toc42357436)

[LAB 2. PROCESS AND THREAD 4](#_Toc42357437)

[LAB 3. DEADLOCKS 5](#_Toc42357438)

[LAB 4. MEMORY 6](#_Toc42357439)

[LAB 5. IO 7](#_Toc42357440)

* FULL NAME: Nguyễn Minh Hiển
* STUDENT ID:DE140105
* GROUP MEMBERS: Minh Hiển
  + - Phi Anh
    - Đức Long
    - Tấn Hưng

# LAB 1. OS INTRO

**Preparation**

1. What will happen if you type man man in Linux?

This command displays the manual page for the MAN program

1. How can you use the command ls to find out about the size of the file */etc/lilo.con*?

The command ls -l/etc/lilo.con will list in long format, giving mode, number of links, owner, group, size in bytes…

3. What happens to the files in the command mv *file1 file2* ? Which option of mv issues a warning?

If both files are in the current directory and I just type `mv file1 file2`,

file1 will write over file2. After this operation, there will be only one file

left, which contain the contents of previous file1 and is named as file2.

The command `mv -i file1 file2` will prompt for confirmation whenever the move

would overwrite an existing target. It looks like

"mv: overwrite file2 (yes/no)?". An affirmative answer means that the move

should proceed. Any other answer prevents mv from overwriting the target.

4. What is the command that you issue if you are in directory / and want to copy the file /mydata to directory /labdata?

cp mydata labdata

5. What is the command that you issue if you are in directory / and want to copy all files and directories under /mydirectory to directory /newdirectory?

The command cp -r mydirectory newdirectory or comannd cp -R /mydirectory

newdirectory will do the job. The option "cp -r" recursively copies the

directory and all its files, including any subdirectories and their files, to

the target. The option "cp -R" is the same as "cp -r", except pipes are

replicated, not read from.

6. What happens when you type rm \* in a directory?

The command `rm \*` will remove all the files in the current directory and will

display the subdirectories that are in the current directory. It looks like

"rm: subdirectory is a directory". The subdirectories won't be removed

7. What is the command used to delete all files and directories under the directory */mydirectory*?

I will use the command `rm -Rf /mydirectory/\*` or `rm -rf /mydirectory/\*`. These

commands recursively remove directories and subdirectories under /mydirectory.

The "-r" or "-R" options recursively remove directories and subdirectories in

the argument list. The option "-f" removes all files (whether write-protected

or not) in a directory without prompting the user

This question covers basic file manipulation. To begin this question use the first button to set up a small file and directory tree in /home/caine. The resulting tree looks like:

/

+--- home

|

+--- caine

|

+--- test1

| +--- file1

| +--- file2

| +--- file3

| +--- file4

|

+--- mydir1

| +--- info1

| +--- info2

|

+--- data

| +--- data1

| +--- data2

|

+--- lines

+--- words

+--- info

8. Bottom of Form

9. COPY file1 from test1 to data. Keep the name as file1.

10. COPY file2 from test1 to data. Change the name as you copy the file to the new name of filecopy1

11. Rename info1 to newinfo1. Do not move it out of mydir1 ?

12. Change directory into mydir1, and then copy "lines" into the current directory.

13. Still in mydir1, concatinate info2 and lines, saving the output as "joined".?

14. Still in mydir1, concatinate info2 and lines and file1 from test1, saving the output as "joined2".?

**Classwork**

**Exercise 1:**

In the console or terminal window, type the following LINUX commands on the command line.

Note and write down the results

$ ls

$ pwd

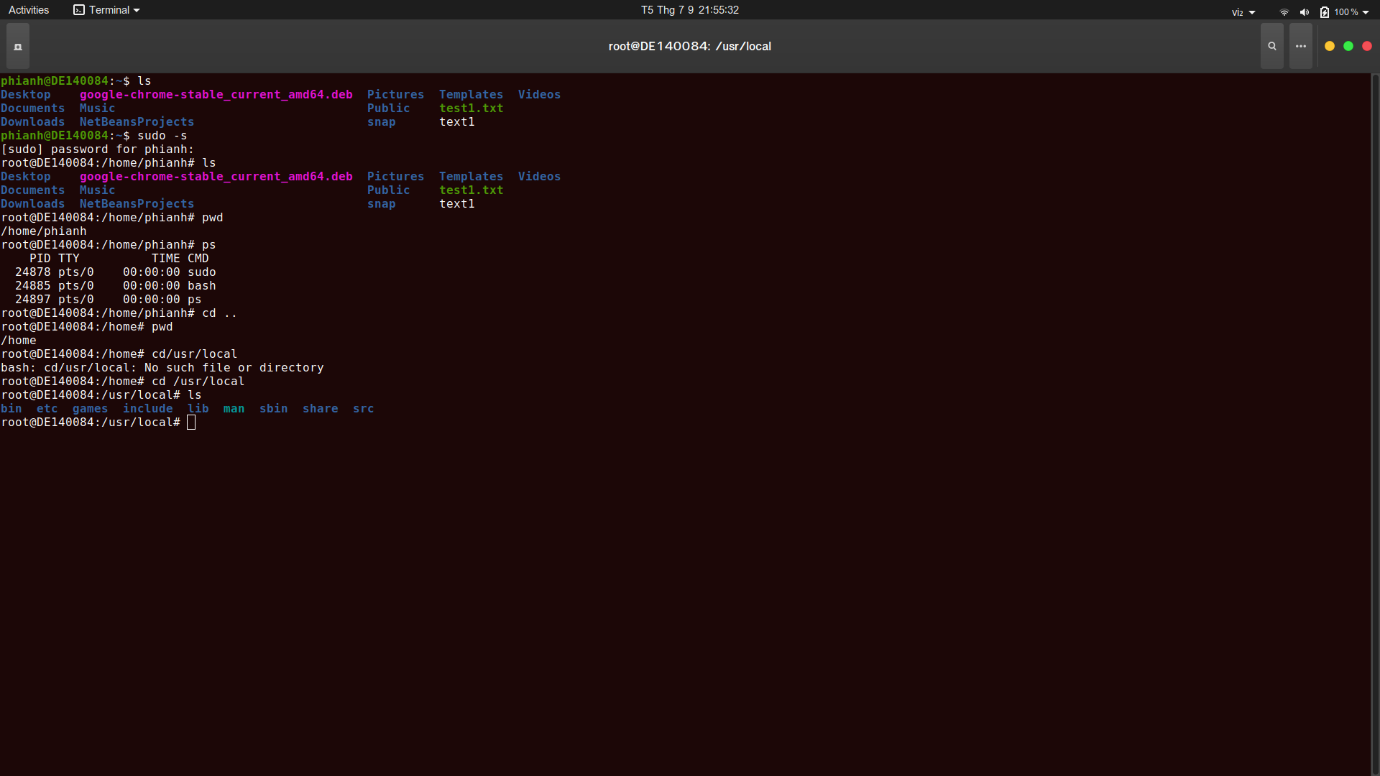
$ps

$ cd ..

$ pwd

$ cd /usr/local

$ ls



**Exercise 2:**

Execute the following command and explain the meaning of each of them.

• ifconfig

• route –n

• hostname

• cat /proc/cpuinfo

• free –m or top

• dpkg ‐l

• uname –r

**Exercise 3:**

In your home folder, create a folder tree as following. Note that, the rectangular describes a folder and the circle represent for a file. You can add any information into your created files.

|  |  |
| --- | --- |
|  |  |

- Create file Hello.java in folder Java and add whatever you want into that file.

- Copy file Hello.java to folder OS and rename it to newHello.txt

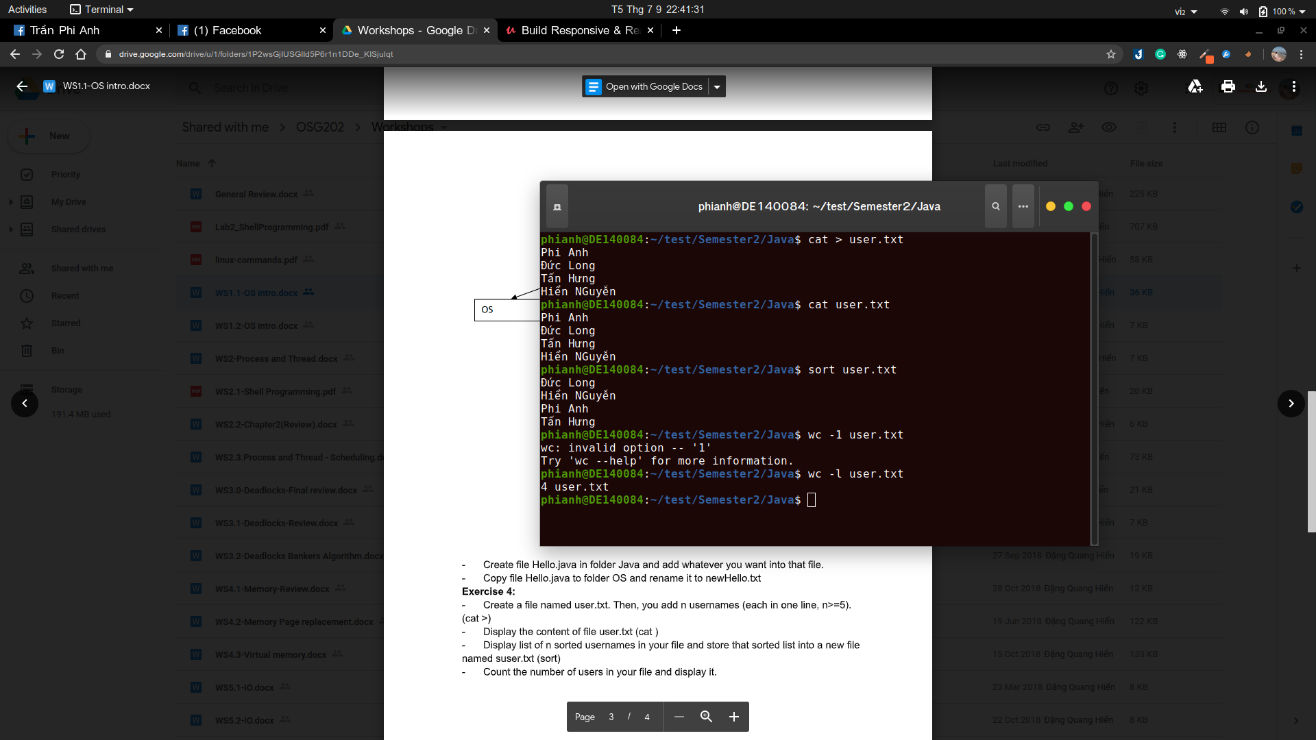
**Exercise 4:**

- Create a file named user.txt. Then, you add n usernames (each in one line, n>=5). (cat >)

- Display the content of file user.txt (cat )

- Display list of n sorted usernames in your file and store that sorted list into a new file named suser.txt (sort)

- Count the number of users in your file and display it.



Q.1 What are the two main functions of an operating system ?

*OS is an extended machine*

* + Hides the messy details which must be performed
  + Presents user with a virtual machine, easier to use

*OS is a resource manager*

* + Each program gets time with the resource
  + Each program gets space on the resource

Q.2 What is multiprogramming?

Multiprogramming is one of the OS features that help the computer run many program at the same time, this happened when user use multiprogram as work, at this time the CPU will switch the progress it’s running to another, because it happened so quickly, we likely suppose the CPU running more than one program at the same time

Q.3 List some differences between personal computer operating systems and mainframe operating systems.

Generally, operating systems for batch systems have simpler requirements than for personal computers. Batch systems do not have to be concerned with interacting with a user as much as a personal computer. As a result, an operating system for a PC must be concerned with response time for an interactive user. Batch systems do not have such requirements. A pure batch system also may have not to handle time sharing, whereas an operating system must switch rapidly between different jobs.

Q.4 What is the key difference between a trap and an interrupt?

-The trap is a signal from user in order to required OS perform specified option immediately

-Interrupt is a signal from the processor announced that event need attention

Q.5 On early computers, every byte of data read or written was directly handled by the CPU (i.e.there was no DMA. What implications does this organization have for multiprogramming ?)

The processor is first assigned to one program. When the program requests to perform an I/O operation, the processor is assigned to another program.

Q6. Which of the following instructions should be allowed only in kernel mode?

(a) Disable all interrupts.

(b) Read the time-of-day clock.

(c) Set the time-of-day dock.

(d) Change the memory map.

Q7. Can the [count = write(fd, buffer, nbytes);] call return any value in *count* other than *nbytes*? If so, why?

If it FAIL count will return -1 otherwise it will return nbytes

Q8. A file whose file descriptor is *fd* contains the following sequence of bytes: 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5. The following system calls are made:

lseek(fd, 3, SEEK\_SET);

read(fd, &buffer, 4);

where the lseek call makes a seek to byte 3 of the file. What does *buffer* contain after the read has completed?

It contains: 1,5,9,2

Q9. A computer uses the relocation scheme of Fig. 1-9(a). A program is 10,000 bytes long and is loaded at address 40,000. What values do the *base* and *limit* register get according to the scheme described in the text?

# LAB 2. PROCESS AND THREAD

**Q0**) What are the possible state transitions of a process?

\_process states:

+ running

+ blocked

+ ready

**Q1**) What are the differences between a thread and a process?

\_A process is a program under execution i.e an active program. A thread is a lightweight process that can be managed independently by a scheduler.

\_Processes require more time for context switching as they are more heavy. Threads require less time for context switching as they are lighter than processes.

**Q2**) What is a race condition?

\_Two processes want to access shared memory at same time and the

final result depends who runs precisely, are called race condition

**Q3**) Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5, and *X*. In what order should they be run to minimize average response time? Given X = 10 and X = 1

We have 5 jobs are waiting to be run P1=9, P2=6, P3=3, P4=5, P5=X:

\_ When X = 10: P3-P2-P4-P1-P5

\_When X = 1: P5-P3-P2-P4-P1

**Q4**) Five batch jobs *A* through *E*, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time.

1. Round robin (RR=4).
2. Priority scheduling.
3. First-come, first-served (run in order 10, 6, 2, 4, 8).
4. Shortest job first.

For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d) assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

**Q5)** What is the difference between preemption and non-preemption in the context of process scheduling.

\_The difference between preemptive and non-preemptive scheduling is that in preemptive scheduling the CPU is allocated to the processes for the limited time. While in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to waiting state

1) What are the possible state transitions of a process?

\_process states:

+ running

+ blocked

+ ready

2) What are the differences between a thread and a process?

\_A process is a program under execution i.e an active program. A thread is a lightweight process that can be managed independently by a scheduler.

\_Processes require more time for context switching as they are more heavy. Threads require less time for context switching as they are lighter than processes.

3) User-level threads vs. Kernel-level threads? Know the pros and cons of each type.

\_User-level threads:

+User-level threads are faster to create and manage

+Implementation is by a thread library at the user level.

+User-level thread is generic and can run on any operating system.

+Multi-threaded applications cannot take advantage of multiprocessing.

**\_**Kernel-level threads:

+Kernel-level threads are slower to create and manage.

+Operating system supports creation of Kernel threads.

+Kernel-level thread is specific to the operating system.

+Kernel routines themselves can be multithreaded.

4) Understand the concepts of race condition and mutual exclusion

+ Locked variables, strict alternation, Peterson's algorithm, disabling interrupts, semaphore, monitor. Know which one is software or hardware solution; which one relies on busy waiting.

+ Semaphores. Understand what a semaphore is; its up and down operations; its principles to allow mutual exclusion among several processes or threads.

\_A semaphore is a variable or abstract data type used to control access to a common resource by multiple processes in a concurrent system such as a multitasking operating system.

+ What is the preemption and non-preemption in the context of process scheduling?

\_ Non-preemptive scheduling algorithm picks process and let it run until it blocks or until it voluntarily releases the CPU

\_ Preemptive scheduling algorithm picks process and let it run for a maximum of fix time

+ How the FIFO, SJF (SJRF) and Round-Robin scheduling algorithms work?

FIFO:

\_Jobs are executed on first come, first serve basis.

\_It is a non-preemptive, pre-emptive scheduling algorithm.

SJF:

Associate with each process the length of its next CPU burst. Use

these lengths to schedule the process with the shortest time

• Two schemes:

– non-preemptive – once CPU given to the process it cannot be preempted until completes its CPU burst

– preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)

• SJF is optimal – gives minimum average waiting time for a given set of processes

Round-Robin:

Each process gets a small unit of CPU time (time quantum), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.

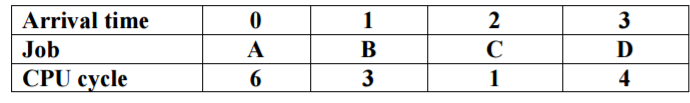
• If there are n processes in the ready queue and the time quantum is q, then each process gets 1/n of the CPU time in chunks of at most q time units at once. No process waits more than (n-1)q time units.

1. Assume that jobs A-D arrive in the ready queue in quick succession and have the CPU cycle requirements listed below. Using the Shortest Remaining Time Next algorithm

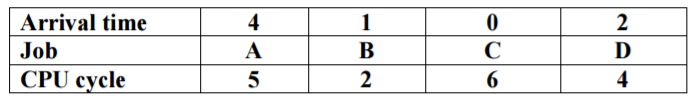
a) The average waiting time is \_\_2.75\_\_

b) The Turnaround time of job D is 6

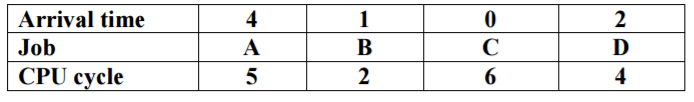
c) The Waiting time of job A is \_\_ 8\_\_.



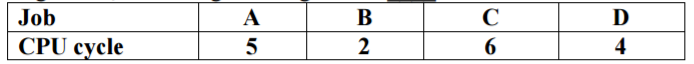
2. A ssume that four jobs A-D require the CPU cycles listed below. Using the Shortest Job First algorithm, the \_C\_ job is run first.



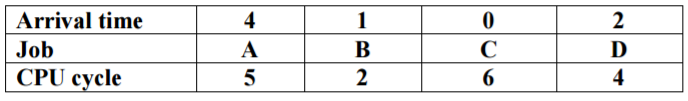
3) A ssume that four jobs A-D require the CPU cycles listed below. Using the Round-Robin algorithm with time slide equaling 4, the average of turnaround time of one process is \_\_\_10.5\_\_\_ .



4) Assume that four jobs A-D require the CPU cycles listed below. Using the Shortest Job First algorithm, the average waiting time is \_\_ 4.75 \_\_. Job



5) A ssume that four jobs A-D require the CPU cycles listed below. Using the Round-Robin algorithm with time slide equaling 4, the waiting time of all process is \_\_\_25\_\_\_ .



6) Assume jobs A-D arrive in quick succession in the READY queue. Using round robin scheduling with time slice equaling 4, the turnaround time for job D is \_\_22\_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **8** | **4** | **9** | **5** |

7) Assume jobs A-D arrive in quick succession in the READY queue. Using shortest job first scheduling, the average turnaround time for each process is \_\_ 14.25 \_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **8** | **4** | **9** | **5** |

8) Assume jobs A-D arrive in quick succession in the READY queue. Using shortest remaining time next scheduling, the average waiting time for each process is \_\_ 6.5 \_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **8** | **4** | **9** | **5** |

9) A ssume that four jobs A-D require the CPU cycles listed below. Using the Shortest Job First algorithm, the \_\_ B \_\_ job is run first.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **5** | **2** | **6** | **4** |

10) Assume jobs A-D arrive in quick succession in the READY queue. Using round robin scheduling with time slice equaling 4, the turnaround time for job C is \_\_\_\_\_\_16\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **4** | **5** | **5** | **4** |

11) Assume jobs A-D arrive in quick succession in the READY queue. Using shortest job first scheduling, the average turnaround time for each process is \_\_\_\_\_ 9.25 \_\_\_\_\_\_ .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **4** | **5** | **5** | **4** |

12) Assume jobs A-D arrive in quick succession in the READY queue. Using shortest remaining time next scheduling, the average waiting time for each process is \_\_\_\_\_ 4.75 \_\_\_\_\_.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **4** | **5** | **5** | **4** |

13) Assume jobs A-D arrive in quick succession in the READY queue. Using round robin scheduling (quantum=4), the average turnaround time for each job is \_\_\_ 18.25 \_\_\_\_\_ .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Arrival time** | **0** | **1** | **2** | **3** |
| **Job** | **A** | **B** | **C** | **D** |
| **CPU cycle** | **8** | **4** | **9** | **5** |

14) Five batch jobs A through E, arrive at a computer center at almost the same time. They have estimated running times of 8, 6, 2, 10, and 4 minutes. Determine the mean process average turnaround time for SJF (Shortest job first) scheduling. Ignore process switching overhead.

14 minutes

# LAB 3. DEADLOCKS

1. A system has four processes and three allocated resources. The current allocation and request needs are as follows:

Process Allocated Currently Request Available

-----------------------------------------------------------------------------------------

A 002 000 100

B 201 202

C 301 001

D 201 100

How many resource instances(in each resource) does that system have after the currently request is provided?

Resources instances is 805.

2. A system has four processes and three allocated resources. The current allocation and request needs are as follows:

Process Allocated Currently Request Available

-----------------------------------------------------------------------------------------

A 002 000 100

B 201 202

C 301 001

D 201 100

What do the processes progress in sequence?

Process Work

A 102

C 403  
D 604

B 805

=> Sequence = ACDB

3. Consider the following state of a system with four processes, P1, P2, P3 and P4, and five types of resources RS1, RS2, RS3, RS4 and RS5 .

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Allocated | Request |  |
| P1 | 01112 | 11021 |  |
| P2 | 01010 | 01021 |  |
| P3 | 00001 | 02031 |  |
| P4 | 21000 | 02110 |  |

Given A = (01021) What do the processes progress in sequence?

R1

R2

R5

R3

R4

4. Consider the following snapshot of a system

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **A** | **B** | **C** | **D** | | **P1** | 0 | 0 | 1 | 2 | | **P2** | 1 | 0 | 0 | 0 | | **P3** | 1 | 3 | 5 | 4 | | **P4** | 0 | 6 | 3 | 2 | | **P5** | 0 | 0 | 1 | 4 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | **A** | **B** | **C** | **D** | | **P1** | 0 | 0 | 1 | 2 | | **P2** | 1 | 7 | 5 | 0 | | **P3** | 2 | 3 | 5 | 6 | | **P4** | 0 | 6 | 5 | 2 | | **P5** | 0 | 6 | 5 | 6 | |  |
| Resources assigned | Resources still requested | E = (3, 14, 12, 12) |

Choose the correct processes using **the deadlock detection algorithm**

5. Assume the following events and actions take place. The following statement\_\_\_\_ is true. Event Action

1) P1 requests and is allocated R1.

2) P2 requests and is allocated R2.

3) P3 requests and is allocated R3.

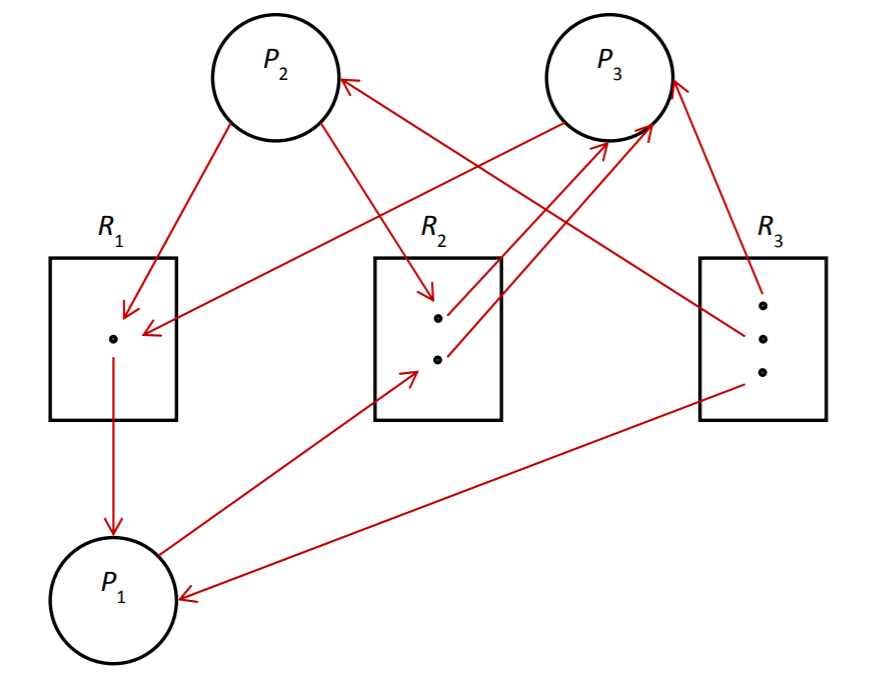
4) P1 requests R2.

5) P2 requests R3.

6) P3 requests R1.

6. A system has three processes (P1, P2, P3) and three reusable resources (R1, R2, R3). There is one instance of R1, two instances of R2 and three instances of R3. P1 holds an R1 and an R3 and is requesting an R2. P2 holds an R3 and is requesting an R1 and an R2. P3 holds two R2 and an R3 and is requesting an R1.

1. Draw the resource allocation graph for this situation.



1. Write all the cycle(s) in the graph.

P3->R1->P1->R2(1st instance)->P3

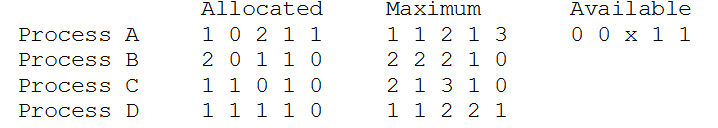
P3->R1->P1->R2(2nd instance)->P3

1. Does a deadlock exist? Why?

Yes, a deadlock does exist.

P3 is waiting for an instance of R1, the only instance of which is held by P1 and P1 is waiting for an instance of R2, both instances of which are held by P3. Since neither an instance of R1 nor that of R2 will ever become available, hence, neither P1 nor P3 will ever be able to continue executing. Therefore, a deadlock exists.

**7. A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:**

****

**What is the smallest value of *x* for which this is a safe state?**

Safe sequence D A C B.

1. What is a deadlock?

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. ... Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).

2. When a deadlock occurs, will the system remain in the deadlocked state permanently or just temporarily for a limited amount of time?

Deadlock is defined as the permanent blocking of a set of processes

that compete for system resources.

3. Can deadlock happen if there is only one process? why?

It is not possible to have a deadlock involving only one single process. The deadlock involves a circular “hold-and-wait” condition between two or more processes, so “one” process cannot hold a resource, yet be waiting for another resource that it is holding.

4. What is the difference btw the coding in Fig.3-2a and Fig.3-2b?

5. Consider Fig. 3-4. Suppose that in step (0) C requested S instead of requesting R and S.

Would this lead to deadlock?

6. Students working at individual PCs in a computer laboratory send their files to be printed by a server which spools the files on its hard disk. Under what conditions may a deadlock occur if the disk space for the print spool is limited? How may the deadlock be avoided?

Disk space on the spooling partition is a finite resource. Every block that comes in de facto claims a resource and every new one arriving wants more resources. If the spooling space is, say, 10 MB and the first half of ten 2-MB

jobs arrive, the disk will be full and no more blocks can be stored so we have a

deadlock. The deadlock can be avoided by allowing a job to start printing before it is fully spooled and reserving the space thus released for the rest of that job. In this way, one job will actually print to completion, then the next one can do the same thing. If jobs cannot start printing until they are fully spooled, deadlock is possible.

7. In the preceding question (Question 6) which resources are preemptable and which are nonpreemptable?

The printer is nonpreemptable; the system cannot start printing another job until the previous one is complete. The spool disk is preemptable; you can  
delete an incomplete file that is growing too large and have the user send it  
later, assuming the protocol allows that.

# LAB 4. MEMORY

1.The problems are we could not have 2 programs running in main memory at once, as this would cause inconsistency in data. Processes could erase each others' written output/input and cause severe inconsistencies.

2.Swapping is a mechanism in which a process can be swapped/moved temporarily out of main memory to a backing store , and then brought back into memory for continued execution.

3.Single contiguous allocation and Partitioned allocation

4.Linked List is Dynamic data Structure .

Linked List can grow and shrink during run time.

Insertion and Deletion Operations are Easier

Efficient Memory Utilization ,i.e no need to pre-allocate memory

Faster Access time,can be expanded in constant time without memory overhead

Linear Data Structures such as Stack,Queue can be easily implemeted using Linked list

5.First fit: In the first fit approach is to allocate the first free partition or hole large enough which can accommodate the process. It finishes after finding the first suitable free partition.

Best fit: The best fit deals with allocating the smallest free partition which meets the requirement of the requesting process. This algorithm first searches the entire list of free partitions and considers the smallest hole that is adequate. It then tries to find a hole which is close to actual process size needed.

Worst fit: In worst fit approach is to locate largest available free portion so that the portion left will be big enough to be useful. It is the reverse of best fit.

Next fit: Next fit is a modified version of first fit. It begins as first fit to find a free partition. When called next time it starts searching from where it left off, not from the beginning

# LAB 5. IO

|  |
| --- |
| **Disk Scheduling Algorithms (5.4.3)**  [FCFS (FIFO)](http://cs.uttyler.edu/Faculty/Rainwater/COSC3355/Animations/diskschedulingfcfs.htm)  [SSTF](http://cs.uttyler.edu/Faculty/Rainwater/COSC3355/Animations/diskschedulingsstf.htm) (Shortest Seek Time First)  SCAN(Scan Algorithm Disk Scheduling)  Elevator(Up/Down) |

|  |  |
| --- | --- |
| Q1 | A disk queue with requests for I/O blocks on cylinders in orders: 10, 22, 20, 2, 40, 6, 38. Assume that the disk head is initially at cylinder 9. How many cylinder do total head movement using **SSF (Shortest Seek First)** algorithms? |
| a. | 47 (true) |
| b. | 69 |
| c. | 45 |
| d. | None of the others |

|  |  |
| --- | --- |
| Q2 | A disk queue with requests for I/O blocks on cylinders in orders: 10, 22, 20, 2, 40, 6, 38. Assume that the disk head is initially at cylinder 38. Which the ordering cylinder in progress do using a slight modification of **elevator algorithms**? |
| a. | 38 38 40 2 6 10 20 22 |
| b. | 38 40 2 6 10 20 22 38 |
| c. | 38 38 22 20 10 6 2 40 |
| d. | 38 38 40 22 20 10 6 2 (true) |

Q.3 Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests, in FIFO order, is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk-scheduling algorithms?

a. FCFS

b. SSTF

c. SCAN

a)The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130. The total seek distance is 7081.

b)The SSTF schedule is 143, 130, 86, 913, 948, 1022, 1470, 1509, 1750, 1774. The total seek distance is 1745.

c)The SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774,4999, 130, 86. The total seekn distance is 9769

1. What is memory mapped I/O?

Memory mapped I/O is a way to exchange data and instructions between a CPU and peripheral devices attached to it. Memory mapped IO is one where the processor and the IO device share the same memory location(memory),i.e.,the processor and IO devices are mapped using the memory address.

1. Why is DMA an improvement over CPU programmed I/O?

Since it operates independently of the CPU, the CPU does not need to manage the process with each transfer. If you are transferring multiple pieces of data, it can be faster using the DMA controller. For I/O, it might be a long time between each transfer.

1. When would DMA transfer be a poor choice?

When we spend more time setting up the DMA transfer than the same operation would take if done entirely by the CPU.

1. Disk Technology. Suppose we have a magnetic disk (resembling an IBM Microdrive) with the following parameters:

Average seek time 12 ms

Rotation rate 3600 RPM

Transfer rate 3.5 MB/second

Sectors per track 64

Sector size 512 bytes

Controller overhead 5.5 ms

Answer the following questions. (Note: you may leave any answer as a fraction.)

(a) What is the average time to read a single sector?

Disk Access Time = seek time + rotational delay + transfer time + controller overhead

rotational delay = rotation rate / 2

Rotation rate = 3600, thus time for one rotation

= 1min \* 60 seconds / 3600 = 0.016second = 16.66ms per rotation

Rotational delay for one disk = 1/ 2 \* 16.66ms = 8.33 = 30.000/RPM = (1 / (RPM / 60)) \* 0.5 \* 1000

Transfer time = (512 / 3.5\*2^20)\*1000 = x ms

= 12 + 8.33 + 0.14 + 5.5 = 25.97ms

b) What is the average time to read 8 KB in 16 consecutive sectors in the same cylinder?

Transfer time = (8 \* 1024 / 3.5\*2^20)\*1000 = x ms

= 12 + 8.33 + 2.24 + 5.5 = 28.07ms

c) Now suppose we have an array of 4 of these disks. They are all synchronized such that the arms on all the disks are always on the same sector within the track. The data is striped across the 4 disks so that 4 logically consecutive sectors can be read in parallel. What is the average time to read 32 consecutive KB from the disk array?

4 disks -> each disk can read a sector at a time, total memory can read at a time = 4 \* 512 (size of sector) = 2KB.

To read 32KB in 4 disks, need to read 8k in each disk 8k = 8 \* 1024 / 512 = 16 sectors

And all disks read at the same time -> answer is same as b

5. What is the average time to read or write a 512-byte sector for a typical disk rotating at 7200 RPM? The advertised average seek time is 8ms, the transfer rate is 20MB/sec, and the controller overhead is 2ms. Assume that the disk is idle so that there is no waiting time. Disk

Access Time = seek time + rotational delay + transfer time + controller overhead

6. A program repeatedly performs a three-step process: It reads in a 4-KB block of data from disk, does some processing on that data, and then writes out the result as another 4-KB block elsewhere on the disk. Each block is contiguous and randomly located on a single track on the disk. The disk drive rotates at 7200RPM, has an average seek time of 8ms, and has a transfer rate of 20MB/sec. The controller overhead is 2ms. No other program is using the disk or processor, and there is no overlapping of disk operation with processing. The processing step takes 20 million clock cycles, and the clock rate is 400MHz. What is the overall speed of the system in blocks processed per second assuming no other overhead?

7. How much cylinder skew is needed for a 7200-rpm disk with a track-to-track seek time of 1 msec? The disk has 200 sectors of 512 bytes each on each track.

8. Disk requests come in to the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. A seek takes 6 msec per cylinder moved. How much seek time is needed for

1. (a) First-Come, first served.
2. (b) Closest cylinder next.
3. (c) Elevator algorithm (initially moving upward).

In all cases, the arm is initially at cylinder 20.

1. The order for FCFS is: 20->10->22->20->2->40->6->38. Distance is  
   10+12+2+18+38+34+32 = 146 cylinders, so time is 146\* 6 = 876 sec.

b)The order for CCN is: 20->20->22->10->6->2->38->40. Distance is  
0+2+12+4+4+36+2 = 60 cylinders, so time is 60 \* 6 =360 msec.

c)The order for elevator is: 20->20->22->38->40->10->6->2. Distance is  
0+2+16+2+30+4+4 = 58 cylinders, so time is 58 \* 6 =348 msec.