GRIFFITH COLLEGE DUBLIN

Class Test

OPERATING SYSTEMS DESIGN

Lecturer: **Dr Faheem Bukhatwa**

Date: April 2018

This is a 75 minute test

ALL QUESTIONS TO BE ATTEMPTED.

ALL QUESTIONS CARRY EQUAL MARKS.

1. What is a semaphore? With the help of an example, describe how a semaphore works.

(20 marks)

2. Explain the priority CPU scheduling algorithm and outline the difference between a pre-emptive priority and a non-pre-emptive priority CPU scheduling algorithm. Give an example.

(20 marks)

3. Describe Bankers algorithm and outline <u>five</u> of its problems.

(20 marks)

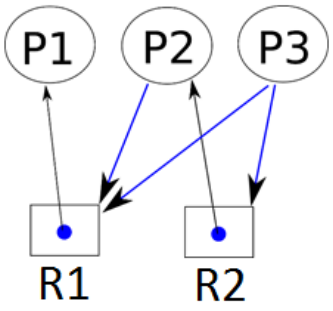
4. In demand paging, a process requests pages in the following order:

   d c b a d c e d c b a e.

   Construct a page trace analysis indicating page faults with an asterisk (*) using First-in-first-out (FIFO) where memory is divided into three page frames.

(20 marks)

5. Explain if the system is deadlocked or not in the following directed resource graph



(20 marks)

1. What is a semaphore? With the help of an example, describe how a semaphore works.
(20 marks)

- A semaphore is a general format of a mechanism used to provide control of access to a resource or a critical part of code through synchronization in parallel programming environment.
- A semaphore is a flag of a non-negative integer variable that's used to control if and when a resource is free and can be accessed by a process in parallel programming.
- An indivisible machine instruction executed in a single machine cycle to see if the key is available and, if it is, sets it to unavailable.
- A binary semaphore is a semaphore which limits access to a critical section at once to one process only.
- A counting semaphore may allow more than one process access to a critical section at once.

Main idea: A process approaching a critical section checks or tests whether semaphore is free or busy (P functions).
If free then in an atomic operation it decrements its value (Test & decrement) and enters the critical section. Else:
If critical section is found busy then the process goes asleep in the provided Q and never checks again.
A process in the critical section, as it leaves, it checks for any waiting processes in the Q waiting (V function). If a process is waiting then it signals it to wake up and enter the critical section. If no waiting processes is in Q then the leaving process increments the variable. That will indicate the critical section is free.

- Value of semaphores is initiated with the number of process that can enter the critical section at once, and can only be changed by the operations:

1. Setting the initial value of the semaphore (number of concurrent accesses allowed).
2. P (*proberen* means to test)
3. V (*verhogen* means to increment)

- If "s" is a semaphore variable, then:
- s = 0 implies busy critical region and the process calling on the P operation must wait until s > 0

P(s): If s > 0 then s: = s – 1
(test, fetch, decrement, and store sequence)
{{
Critical section
}}
V(s): s: = s + 1
(fetch, increment, and store sequence)

- Choice of which of the waiting jobs will be processed next depends on the algorithm used by this portion of the Process Scheduler

2. Explain the priority CPU scheduling algorithm and outline the difference between a pre-emptive priority and a non-pre-emptive priority CPU scheduling algorithm. Give example.

(20 marks)

Non-preemptive Priority scheduling is an algorithm that is commonly used in batch systems.

- Gives priority to more important jobs.
- Allows jobs with the highest priority to be processed first.
- Jobs are not interrupted until their CPU cycles are completed or a natural wait occurs.
- If two jobs with equal priority are in the READY queue, priority is given to the job that arrived first.
- Jobs usually linked to one of several READY queues, based on their priority.
- Priorities can either be assigned by the System Administrator based on characteristic extrinsic to the jobs (e.g. user group) or by the Process Scheduler based on characteristics intrinsic to the jobs (e.g. memory requirements, total CPU time, amount of time already spent in the system).
- Jobs are assigned to queues based on their priority – can have multiple queues, with a different queue for each priority level.
- The priority assigned to a job can be adjusted while the job is in the system.
- A job can also have its priority increased, e.g. when it issues an I/O request before its time quantum has expired.

- **N0n-Preemptive priority scheduling**: When a job arrives that has a higher priority than the job using the CPU, No interrupt occurs. The running job using CPU continues to do so.
- **Preemptive priority scheduling**: When a job arrives that has a higher priority than the job using the CPU, an interrupt occurs. The running job is preempted (interrupted) and moved to the ready state.

- **Priority scheduling ensures fast completion of important jobs but results in infinite postponement of some jobs**

solution

- **Adjusts the priorities assigned to each job**
- **A job may also have its priority increased**
- **Good in interactive systems**
-

3. Describe Bankers algorithm and outline <u>five</u> of its problems.

(20 marks)

Description 10 marks

- Deadlock can be avoided if system knows ahead of time sequence of requests associated with each of the active processes
- When a system receives a request, the system plays the What-if this request is satisfied then will it leave the system in an Un-safe state? If it does then the request is denied.

Dijkstra's Bankers Algorithm (1965) regulates resources allocation to avoid deadlock
No customer granted loan exceeding bank's total capital

All customers given a maximum credit limit
No customer allowed to borrow over the limit
The sum of all loans won't exceed bank's total capital

- To avoid deadlock, OS must make sure:

Never satisfy a request that moves it from a safe state to an unsafe one
Must identify the job with the smallest number of remaining resources
Number of available resources is always equal to, or greater than, the number needed for the selected job to run to completion

5 problems 3 marks each

- Problems with Banker's Algorithm:

1. Jobs must state the maximum number of resources needed
2. Number of total resources for each class must remain constant
3. Number of jobs must remain fixed
4. Overhead cost incurred can be quite high
5. Resources aren't well utilized because the algorithm assumes the worst case
6. Scheduling suffers as a result of poor utilization and jobs are kept waiting for resource allocation

4. In demand paging, a process requests pages in the following order:

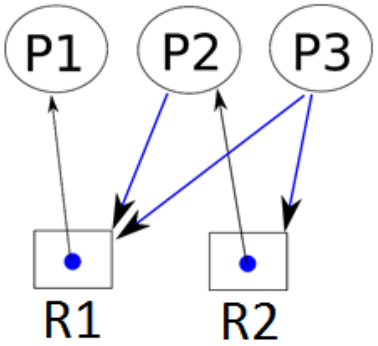
d c b a d c e d c b a e.

Construct a page trace analysis indicating page faults with an asterisk (*) using First-in-first-out (FIFO) where memory is divided into three page frames.

(20 marks)

Page Request	**d**	**c**	**b**	**a**	**d**	**c**	**e**	**d**	**c**	**b**	**a**	**e**
Page frame 1	d	d	d	a	a	a	e	e	e	e	e	e
Page frame 2		c	c	c	d	d	d	d	d	b	b	b
Page frame 3			b	b	b	c	c	c	c	c	a	a
Page Fault	*	*	*	*	*	*	*			*	*	

5. Explain if the system is deadlocked or not in the following directed resource graph



(20 marks)

No deadlock
P1 has all resources it needs. It finishes. It releases R1
R1 is allocated to P2.
P2 has all resources it needs. It finishes. It releases R1, R2
R1 and R2 are allocated to P3.
P3 has all resources it needs. It finishes. It releases R1, R2

6) Explain the operation of SPOOLING?

(20 marks)

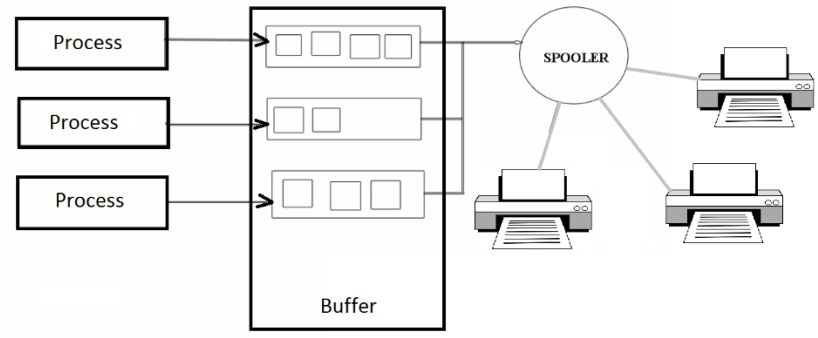
SPOOLING (Simultaneous Peripheral operation On-Line)

One of the earliest applications of multiprogramming spooling gives the ability to avoid delays caused by slow speed devices such as terminals, printers etc. It is also a techniques which transforms a dedicated device into what appears to be a shared device, known as virtual device. Spooling is a process where sent data is temporary placed in a buffer temporary.

Data sent to be printed from multiple processes is accepted and written to a buffer (or an external hard disk). The buffer is much faster and process can continue processing using CPU. This provides much faster use of CPU. When a job finishes its printing, or when the job terminates then all its printed out put is directed to the printer. It is quite normal for the current printing output on the printer belongs to a job which has terminated quite a while back.

For input, the OS reads data from the slow input device, stores it to a fast device or to a buffer in memory. Any further requests of input the OS takes the data from the buffer.
On output, data is written to fast media or a buffer in memory enabling the program to believe it is finished and exit allowing the OS to load and execute another process. The output is then processed from the buffer to the final output device, (typically a printer). Hence, one process can be executing while the printer is printing the output of a previous process.

Originally, spooling buffered output on the same device in a multiprogramming environment.



A spooler has a similar idea to and has become to be a print manager or a print server, queuing print jobs for printers.