Operating systems I

(1DT044)

Operating systems and process-oriented programming

(1DT096)

Final written exam

Friday 2018-03-16, Fyrishov, 14:00 - 19:00

Correct answers

Mixed concepts

L_C	Concept
A	System call
B	Critical section
C	Paging
D	TLB
E	Context switch
F	Round Robin
G	SJF
H	Many-to-one
I	fork
J	exec
K	wait
L	Operating systems
M	Message-passing
N	Pipe
O	Signal
P	Race condition
Q	Peterson's solution
R	Throughput
S	Response time
T	External fragmentation

L_S	Statement
K	Suspends the execution of the parent process while the child executes.
O	A notification sent to a process in order to notify it of an event that occurred.
G	A non-preemptive scheduling algorithm.
	Sits between the main memory and the CPU registers.
	A wrapper around the command interpreter that adds useful features that makes it easer to enter commands.
H	Entire process will block if a thread makes a blocking system call.
L	Controls the hardware and coordinates its use among the various application programs for the various user.
B	Requires mutual exclusion.
Q	A concurrent programming algorithm for mutual exclusion.
C	Solves the problem with external fragmentation.
S	Amount of time it takes from when a request was submitted until the first response is produced.
A	Requesting service from the kernel of the operating system.
I	A process creates a copy of itself.
T	Total memory space exists to satisfy a request, but it is not contiguous.
D	Improves virtual address translation speed.
E	Enables multiple processes to share a single CPU and is an essential feature of a multitasking operating system.
N	A simplex FIFO communication channel that may be used for one-way interprocess communication (IPC).
R	Number of processes that complete their execution per time unit.
P	Behaviour of an electronic, software or other system where the output is dependent on the sequence or timing of other uncontrollable events.
	Requires a priori information.
F	Assigns a fixed time unit per process, and cycles through them.
J	Runs an executable file in the context of an already existing process.
M	Useful for exchanging smaller amounts of data, because no conflicts need be avoided.
	A variation on linked allocation.

Module 1

1.1) B

1.2) D

1.3) A

1.4) C

1.5) C

Module 2

2.1) D

2.2) 0 = New, 1 = Ready, 2 = Running, 3 = Terminated, 4 = Waiting

2.3) a = fork, b = exec, c = wait, d = exit

2.4) a = I/O request completion, b = I/O request, c/d = time slice interrupt, c/d = fork()

2.5) D

2.6) A

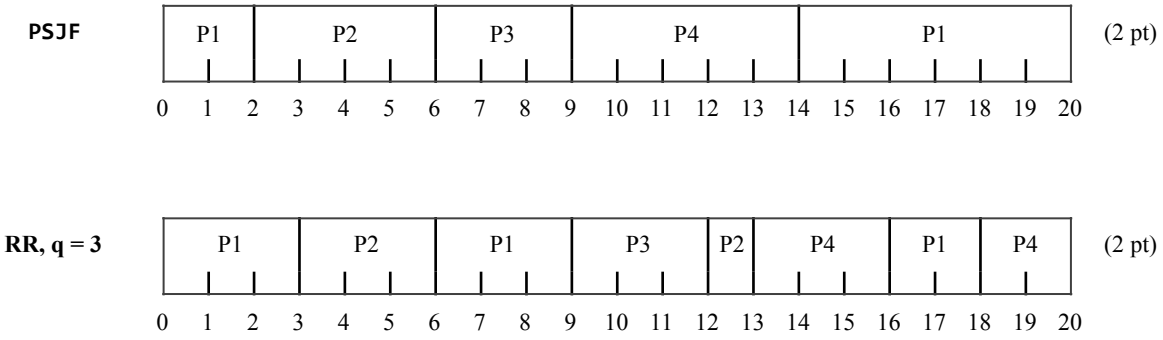
Module 3

3.1) C

3.2) D

3.3) Average response time = 5/4 = 1.25, Average waiting time = 30/4 = 7.5

3.4)



Module 4

4.1) B
4.2) Mutual exclusion, No preemption, Hold and wait, Circular wait
4.3) D
4.4) C
4.5)

	Allocation				Max				Need				
Task	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**	**Done**
T_0	4	1	0	0	6	5	6	0	2	4	6	0	TRUE
T_1	2	3	6	0	2	5	6	0	0	2	0	0	TRUE
T_2	4	5	3	1	6	5	3	2	2	0	0	1	TRUE
T_3	0	0	0	1	0	5	7	1	0	5	7	0	TRUE
T_4	2	1	0	0	2	1	0	0	0	0	0	0	

	Available				
Step	**A**	**B**	**C**	**D**	**Choice**
1	0	2	5	1	T1
2	2	5	11	1	T0
3	6	6	11	1	T2
4	10	11	14	2	T3
5	10	11	14	3	T4
-	12	12	14	3	-

Yes, the state is safe as we found a sequence <T1, T0, T2, T3, T4> that made it possible for all tasks to be grated all their needs at once.

Other similar sequences are also possible. For example by starting with T3 or T4. For all sequences the amount of available resources at the end must be [12, 12, 14, 3].

Module 5

5.1) C

5.2) Virtual address space = 4 GiB, Physical address space = 256 MiB, page/fram size = 4 KiB

5.3) A

5.4) B