

COMP3230B Principle of Operating Systems

2020 Midterm Examination

(Answer Book)

You should write all your answers in the answer book and submit it to Moodle before 11:59AM Oct 24, 2020. The acceptable file formats are pdf and MS Words. The file name should be named as COMP3230B-YourUID-LastName-FirstName (e.g., COMP3230B-1234567-Xu-Pengfei.pdf). DON'T take photos via a mobile phone and embed the photos to the Words file, nor using a Scan App to produce the pdf file for the submission.

Name: _____ Ngai Cheuk Hin _____

University Number: _____ 3035488160 _____

Curriculum: CE/BBAIS/CS/EE, or other (QFin, exchanged) _____ BBAIS _____

Problem 1	/ 45
Problem 2	/ 15
Problem 3	/ 20
Problem 4	/ 20
Total	/ 100

Problem 1. Multiple Choice Questions (45 marks)

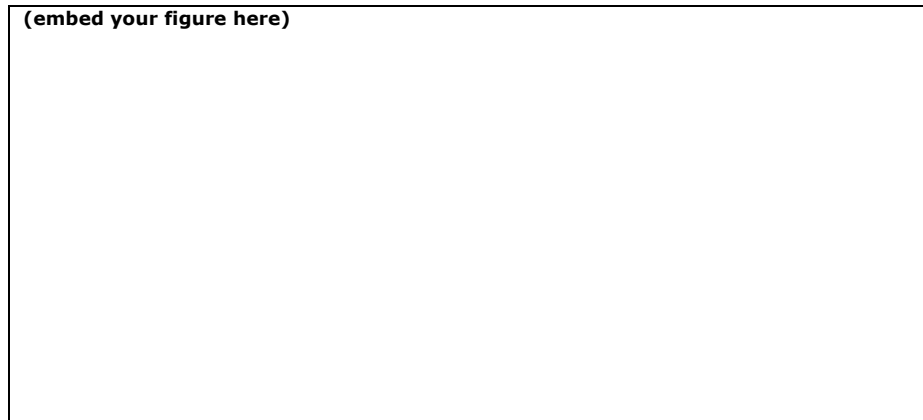
Your choice(s)	Correct each wrong statement here
1. (1, 2, 3, 4)	<p>1. physical cores -> logical cores</p> <p>2. double the processing speed -> 30% increase in the processing speed</p> <p>3. a single user -> all users</p> <p>4. all user-space process -> specific user's user-space process</p>
2. (1)	<p>1. is much faster -> is much slower</p>
3. (4)	<p>4. virtual machine -> uncompressed Linux kernel</p>
4. (1, 3)	<p>1. directly executable -> a part of executable</p> <p>3. calling process' stack segment -> memory mapped region for shared libraries</p>
5. (4)	<p>4. VIRT and RES -> Only VIRT</p>
6. (1, 2, 3, 4)	<p>1. user defined -> OS provided</p> <p>2. system call functions -> wrapper functions</p> <p>3. system call number -> interrupt factor</p> <p>4. less than -> more than</p>
7. (2, 3, 4, 5)	<p>2. data -> stack</p> <p>3. stack -> BSS</p> <p>4. heap -> data</p> <p>5. heap segment -> somewhere in the memory allocated by OS</p>
8. (2, 4)	<p>4. heap -> text, data, BSS</p>
9. (3, 4)	<p>3. same virtual address but different physical address -> different virtual address but same physical address</p> <p>4. parent process -> OS</p>

10. (3)	3. all child -> a child
11. (1, 2, 3, 4)	1. zombie process -> orphan process 2. zombie process -> orphan process 3. memory -> space in process table 4. [zombiePID] -> [parentPID of zombie process]
12. (3)	3. SJF -> FIFO
13. (1, 3, 4)	1. HRRN -> SRTF 3. HRRN -> SJF and SRTF 4. remaining service time -> required service time
14. (1, 2, 4)	1. kill() -> kill(pid, SIGTERM) 2. signal() -> kill() 4. from a parent process to a child process -> from a child process to a parent process
15. (1, 3)	1. global queue -> Multilevel Feedback Queue Scheduling(MFS) + Completely Fair Scheduler (CFS) 3. taskset 0x7 [pid] -> taskset -p 0x80 [pid]

Problem 2: Process Creation using Fork() (15 marks)

You are allowed to draw a figure using any drawing tools (e.g., Paint, PowerPoint) to illustrate your ideas if it helps (Not Compulsory):

(embed your figure here)



Questions	Your answers
(a) 7%	<p>2 times.</p> <p>1 time from the Parent process. 1 time from the Child process in the <code>pid1 = fork()</code>; Grandchild process would not print as it exits before print.</p>
(b) 8%	<p>2.</p> <p>Parent process and Child process both get the count to be 1 after the 1st fork. Then, Child process increase the count to 2 after it forks. Count is increase to 12 in the Grandchild process but it exits after the count increases to 12 but before print. Therefore, count would not be displayed as 12. Then in the else condition, the count in the Parent process increases to 2.</p> <p>Finally, Parent process and Child process both print count as 2.</p>

Problem 3: Signals and Shared Memory (20%)

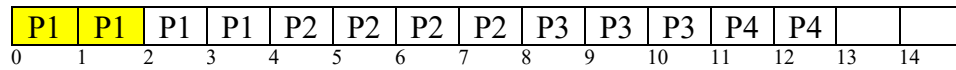
Answer:

Code segment	C code to be added by you (or leave it blank if nothing is needed)
A	// initialization shmids = shmget(IPC_PRIVATE, 10*sizeof(int), IPC_CREAT 0666);
B	// child: b = (int *)shmat(shmids,0,0);
C	// child: for-loop kill(getpid(), SIGSTOP);
D	// child: after for-loop shmdt(b);
E	// parent a = (int *)shmat(shmids,0,0);
F	// parent: for-loop (before a[i]= i) waitpid(pid, &status, WUNTRACED);
G	// parent: for-loop (after printf()) kill(pid, SIGCONT);
H	// before the end of the program shmdt(a);

	<code>shmctl(shmid, IPC_RMID, NULL);</code>
Analysis Part (6%)	<p>Child process use <code>kill(getpid(), SIGSTOP)</code> to send stop signal to itself in each iteration.</p> <p>On the other hand, Parent process uses <code>wait(pid, &status, WUNTRACED);</code> to wait and check whether the Child process is stopped. It returns and run the following code if Child process is stopped. After it finished writing and printing, it sends the signal to Child by <code>kill(pid, SIGCONT);</code> to inform Child to print.</p> <p>Both processes use <code>shmdt()</code> function to de-attach the shared memory before they terminates. Also, <code>shmctl(shmid, IPC_RMID, NULL);</code> would mark the shared memory segment to be destroyed before Parent terminates.</p>

Problem 4: Process Scheduling (20 marks)**FIFO: (6%)**

(2%) Fill up the Gantt chart (You can use different background colors to represent different processes. Not Compulsory)

**(2%) fill up the table**

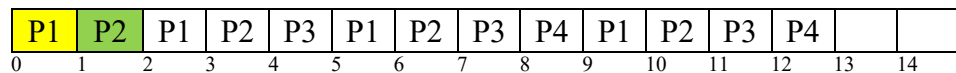
Process ID (PID)	Arrival Time	CPU Time	Completion Time	Turn Around Time	Waiting Time
1	0	4	4	4	0
2	1	4	8	7	3
3	4	3	11	7	4
4	8	2	13	5	3

(1%) Average Turn Around time = 5.75

(1%) Average Waiting time = 2.5

RR: (6%)

(2%) Fill up the Gantt chart. (You can use different background colors to represent different processes. Not Compulsory)

**(2%) fill up the table**

Process ID (PID)	Arrival Time	CPU Time	Completion Time	Turn Around Time	Waiting Time
1	0	4	10	10	6
2	1	4	11	10	6
3	4	3	12	8	5
4	8	2	13	5	3

(1%) Average Turn Around time = 8.25

(1%) Average Waiting time = 5

SRTF: (8%)

(4%) Fill up the Gantt chart. (You can use different background colors to represent different processes. Not Compulsory)

P1	P1	P1	P1	P3	P3	P3	P2	P4	P4	P2	P2	P2		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

(2%) fill up the table

Process ID (PID)	Arrival Time	CPU Time	Completion Time	Turn Around Time	Waiting Time
1	0	4	4	4	0
2	1	4	13	12	8
3	4	3	7	3	0
4	8	2	10	2	0

(1%) Average Turn Around time = 5.25

(1%) Average Waiting time = 2

---- END of the Answer Book ----