Ph.D. Qualifiers Exam Spring 2020
Operating Systems

Computer Science Department,
New Mexico State University

Exam time: 120 min. The exam contains a total of 6 problems, all equal weight.

If you believe that you cannot answer a question without making some assumptions, state those assumptions in your answer.

Irrelevant verbosity will not gain you points. Clear and crisp answers will be appreciated.

This is a closed-book, closed-note exam.

Textbook:
Operating Systems Concepts, Ninth Edition, by Silberschatz, Galvin, and Gagne published by John Wiley and Sons.

Qn1. Deadlock is a potentially nasty problem one encounters in multi-threaded programs.

1. Please list the necessary conditions for a deadlock to occur.

   Mutual Exclusion, Hold and Wait, No Preemption, Circular Wait

2. With deadlock prevention, the system ensures that deadlock does not occur by preventing one of these conditions from holding. Match each of the following techniques with the one deadlock condition that it prevents. Explain your choice.

   - Impose a total ordering (or ranking) on how resources are acquired
     prevents Circular Wait condition
   - When a process requests a resource that is already held, force the process holding the resource to release it
     prevents No Preemption condition
   - Only allow a process to request a resource when the process has none
     prevents Hold and Wait condition
   - Allow all processes to access the resource simultaneously
     prevents Mutual Exclusion condition
   - Require each process to grab all desired resources at once
     prevents Hold and Wait condition

Textbook source: Chapter 7.4 from page 323 to 325

Qn2. Consider the following program:

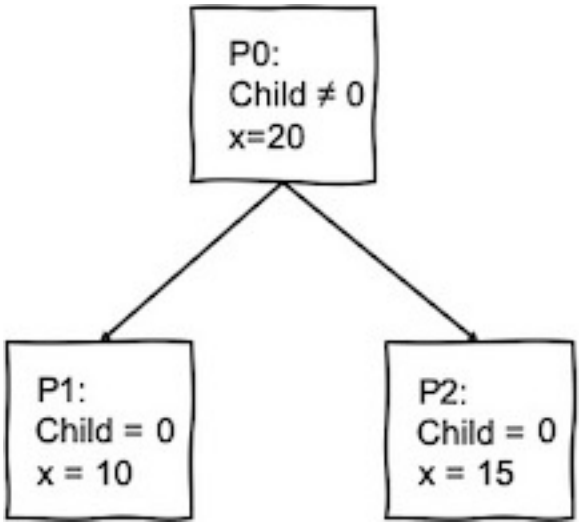
```
main (int argc, char **argv) {
        int child = fork();
        int x = 5;

        if (child == 0) {
                x+=5;
        }
         else {
                child = fork();
                x+=10;
                if (child !=0){
                x+=5;
                }
        }
}
```

1. What is the return value of fork()?

   The function fork returns an integer equal to 0 to the child process and one different from 0 to the parent process.

2. How many processes are created, including the parent process? Draw a simple tree diagram to show the parent-child hierarchy of the spawned processes.



3. How many different copies of the variable x are there? What are their values when their process finishes?.

   Three copies, x=20, x=10, x=15

Textbook source: Chapter 3.3 from page 116 to 119

Qn3. Consider the following snapshot of a system:

	Allocation				*Max*				*Available*			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

1. What is the content of the *Need* matrix?
2. Is the system in a safe state? If the state is safe, illustrate the order in which the processes may complete.
3. If a request from process P_1 arrives for $(0, 4, 2, 0)$, can the request be granted immediately, why?

1.

Need			
A	**B**	**C**	**D**
0	**0**	**0**	**0**
0	**7**	**5**	**0**
1	**0**	**0**	**2**
0	**0**	**2**	**0**
0	**6**	**4**	**2**

2. The system is in a safe state. Safe order is $< P_0, P_2, P_1, P_3, P_4 >$.

3. The request can be granted immediately, because the resulting state is safe.

Textbook source: Chapter 7.5 from page 331 to 333

Qn4. Consider a system with 3 physical frames of memory that is given the following page memory reference sequence:

1, 3, 6, 7, 1, 3, 6, 7, 1, 3, 6, 7

What is the number of page faults that would occur for each of the following page replacement algorithms?

1. An optimal page replacement algorithm

   6 page faults: 1; 1 3; 1 3 6; 1 3 7; 1 6 7; 3 6 7

2. LRU

   12 page faults: 1; 1 3; 1 3 6; 7 3 6; 7 1 6; 7 1 3; 6 1 3; 6 7 3; 6 7 1; 3 7 1; 3 6 1; 3 6 7

3. Does an optimal page replacement algorithm exist that does not require future knowledge for cyclical memory reference sequences such as shown above? If so, give a general algorithm. If not, explain why.

   Yes, MRU, that discards the most recently used items first can achieve the same performance as the optimal algorithm

Textbook source: Chapter 9.4 from page 414 to 416

Qn5. Consider a computer system with a 32-bit virtual address, 4K page size, and 4 bytes per page table entry.

- How many pages are in the virtual address space?
- What is the maximum size of addressable physical memory?
- If the size of a process is 4GB, which page table would you like to use, one-level, two-level, or three-level? Why?

1. $2^{32}/2^{12} = 2^{10}$ **pages**
2. **With 4 byte page table entry we can reference 2^{32} pages. The maximum addressable physical memory size is $2^{32} * 2^{12} = 2^{44}$ bytes**
3. **Since each table entry is 4 bytes, we can fit at most 2^{10} entries in one page. Two-level page table can be used,**

10 bits	**10 bits**	**12 bits (page offset)**

Textbook source: Chapter 8.6 from page 378 to 380

Qn6. Apple originally did not include support for multi-tasking in its iphone OS. Why do you think they made this design decision? The newer iphone OS, however, includes support for multi-tasking. What were the technical arguments that might have prompted Apple engineers to make this change? What (if any) problems might this give to the user of the device?

- **It's an open question.The original IOS may be limited by its memory size, CPU capability or battery life. Single-tasking design is more reliable and effective. The newer iphone comes with larger memory and higher computational ability, which is able to support sophisticated OS structure.**
- **To support multi-tasking, system may need large memory, sophisticated CPU schedulers, complex frame allocation and page replacement algorithms, protection mechanisms etc. to efficiently manage all processes and avoid possible deadlock and thrashing.**

Textbook source: Chapter 3.2 from page 111 to 115