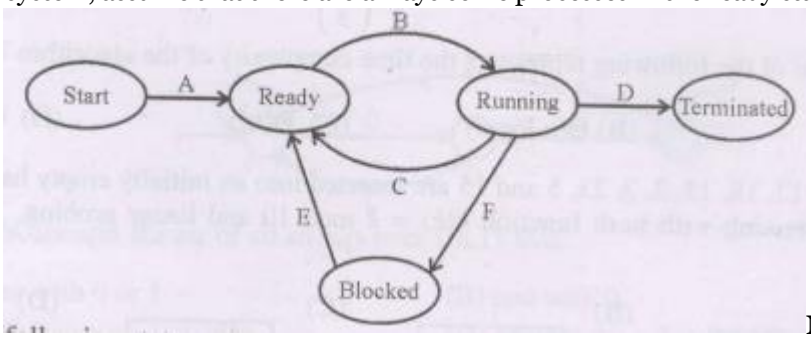
Home work – Set 3

1. Answer yes/no, and provide a brief explanation.
   (a) Can two processes be concurrently executing the same program executable?
   (b) Can two running processes share the complete process image in physical memory (not justparts of it)?

2. What are the properties inherited by a child process from parent process?

3. In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state:



Now consider the following statements:
   I. If a process makes a transition D, it would result in another process making transition A immediately.
   II. A process P2 in blocked state can make transition E while another process P1 is in running state.
   III. The OS uses preemptive scheduling.
   IV. The OS uses non-preemptive scheduling.

   Which of the above statements are TRUE?
   (i) I and II
   (ii) I and III
   (ii) II and III
   (iv) II and IV

4. Consider a process executing on a CPU. Give an example scenario that can cause the process to undergo:
   (a) A voluntary context switch.
   (b) An involuntary context switch.

5. Consider a parent process P that has forked a child process C. Now, P terminates while C is still running. Answer yes/no, and provide a brief explanation.
   (a) Will C immediately become a zombie?
   (b) Will P immediately become a zombie, until reaped by its parent?

6. Which of the following actions by a running process will always result in a context switch of the running process, even in a non-preemptive kernel design?
   (a) Servicing a disk interrupt, that results in another blocked process being marked as ready/runnable.
   (b) A blocking system call.
   (c) The system call exit, to terminate the current process.1
   (d) Servicing a timer interrupt.

7. Consider a parent process P that has forked a child process C in the program below.

```
int a = 5;
int fd = open(...) //opening a file
int ret = fork();
if(ret >0) {
close(fd);
a = 6;...}
else if (ret == 0) {
  printf("a=%d\n", a);
  read(fd, something);
}
```

After the new process is forked, suppose that the parent process is scheduled first, before thechild process. Once the parent resumes after fork, it closes the file descriptor and changes thevalue of a variable as shown above. Assume that the child process is scheduled for the first timeonly after the parent completes these two changes.

(a) What is the value of the variableaas printed in the child process, when it is schedulednext? Explain.

(b) Will the attempt to read from the file descriptor succeed in the child? Explain.

8. Consider a parent process that has forked a child in the code snippet below.

```
int count = 0;
ret = fork();
if (ret == 0) {
  printf("count in child=%d\n", count);
}
else {
  count = 1;
}
```

The parent executes the statement ”count = 1” before the child executes for the first time. Now,what is the value of count printed by the code above? Assume that the OS implements a regularfork (not a copy-on-write fork).

9. Repeat the previous question for a copy-on-write fork implementation in the OS. Recall thatwith copy-on-write fork, the parent and child use the same memory image, and a copy is madeonly when one of them wishes to modify any memory location.

10. Consider the wait family of system calls (wait, waitpid etc.) provided by Linux. A parent process uses some variant of the wait system call to wait for a child that it has forked. Which of the following statements is always true when the parent invokes the system call?

(a) The parent will always block.
(b) The parent will never block.
(c) The parent will always block if the child is still running.
(d) Whether the parent will block or not will depend on the system call variant and the optionswith which it is invoked.

11. Consider a simple linux shell implementing the command ’sleep 100’. Which of the followingis an accurate ordered list of system calls invoked by the shell from the time the user enters this command to the time the shell comes back and asks the user for the next input?

(a) wait-exec-fork
(b) exec-wait-fork3
(c) fork-exec-wait
(d) wait-fork-exec

12. Consider a process that has requested to read some data from the disk

and blocks. Subsequently,the data from the disk arrives and the interrupt is serviced. However, the process doesn't start running immediately. What is the state of this process at this stage?

13. Consider a process P1 that forks P2, P2 forks P3, and P3 forks P4. P1 and P2 continue toexecute while P3 terminates. Now, when P4 terminates, which process must wait for and reapP4

14. Consider the following lines of code in a program running on xv6.

```
int ret = fork();
if (ret == 0) { //do something in child}
else { //do something in parent}
```

(a) When a new child process is created as part of handling fork, what does the kernel stack of the new child process contain, after fork finishes creating it, but just before the CPU switches away from the parent?

(b) How is the kernel stack of the newly created child process different from that of the parent?

(c) The EIP value that is present in the trap frames of the parent and child processes decides where both the processes resume execution in user mode. Do both the EIP pointers in the parent and child contain the same logical address? Do they point to the same physical addressin memory (after address translation by page tables)? Explain.

(d) How would your answer to (c) above change if xv6 implemented copy-on-write during fork?

(e) When the child process is scheduled for the first time, where does it start execution in kernelmode? List the steps until it finally gets to executing the instruction after fork in the program above in user mode

15. A stack does not contain
(i) function parameters
(ii) local variables
(iii) return addresses
(iv) PID of child process