CSC 4420
Computer Science Operating Systems

Due 5/30/2022 11:59pm

Homework 1

25 points – 8 questions

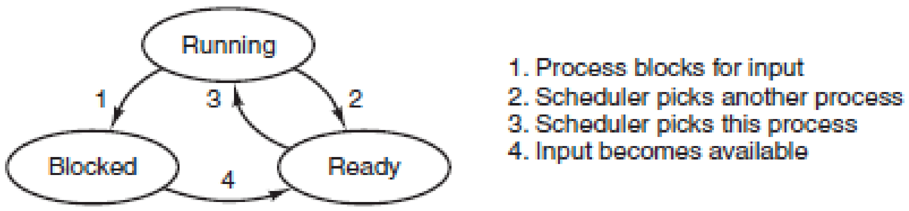
Question 1 (3 points): What is the difference between timesharing and multiprogramming systems? Use your own words (do not copy and paste from slides!).

Question 2 (3 points): Instructions related to accessing I/O devices are typically privileged instructions, that is, they can be executed in kernel mode but not in user mode. Give a reason why these instructions are privileged.

Question 3 (3 points): Cache coherency: Explain what it is and why the OS needs to ensure it.

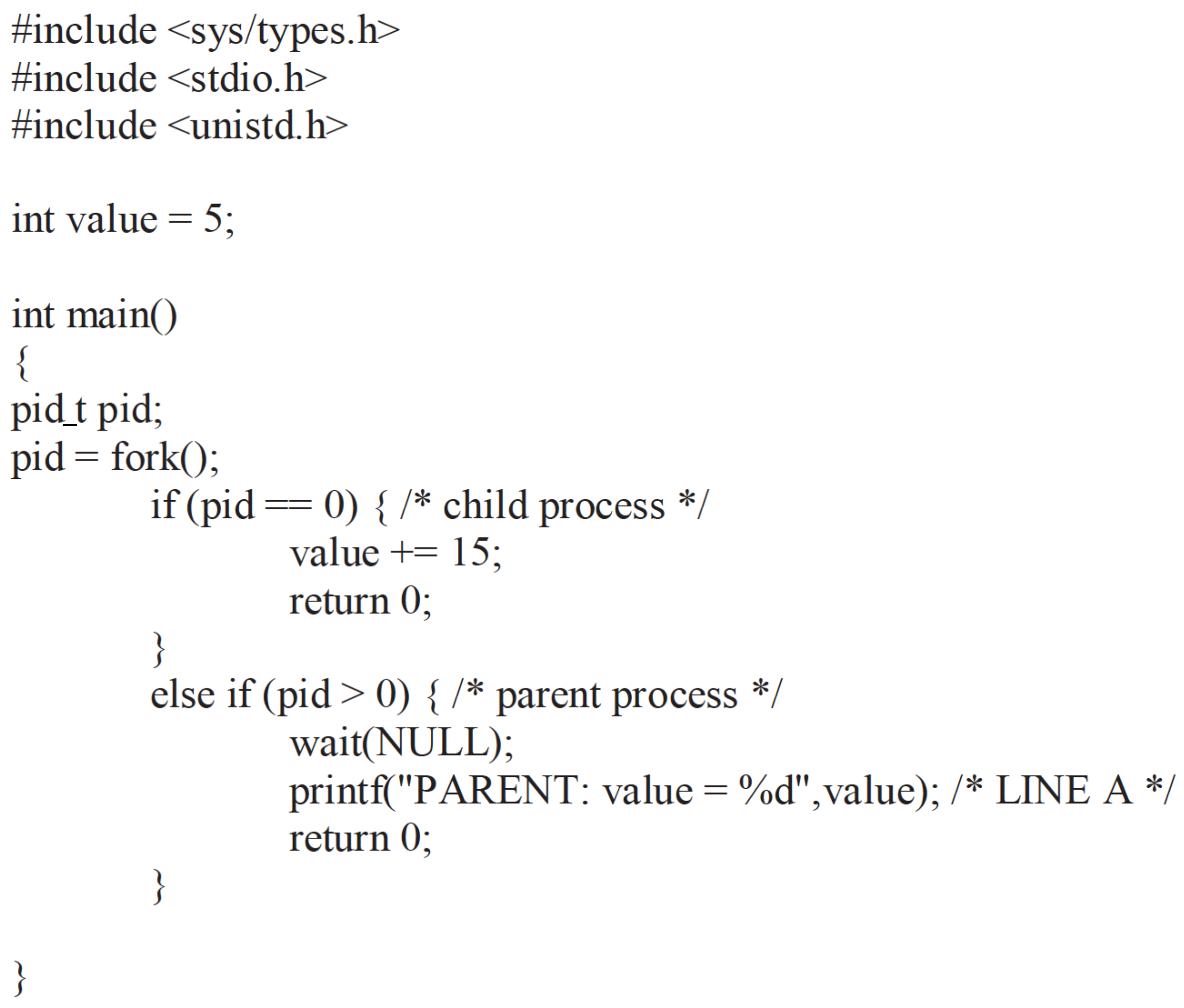
Question 4 (3 points): On early computers, every byte of data read or written was handled by the CPU (i.e., there was no DMA). What implications does this have for multiprogramming?

Question 5 (3 points): In the following figure, three process states are shown. In theory, with three states, there could be six transitions, two out of each state. However, only four transitions are shown. Are there any circumstances in which either or both of the missing transitions might occur?



Running
Blocked
Ready
1
2
3
4
1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Question 6 (4 points): Using the following program, explain what the output will be at LINE A. Motivate your answer.



```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int value = 5;

int main()
{
pid_t pid;
pid = fork();
        if (pid == 0) { /* child process */
                value += 15;
                return 0;
        }
        else if (pid > 0) { /* parent process */
                wait(NULL);
                printf("PARENT: value = %d",value); /* LINE A */
                return 0;
        }

}
```

Question 7 (4 points): Given the code below, what are the values of pid/pid1 at lines A, B, C, and D? Assume that the parent's pid is 10 and the child's pid is 2.

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main(){
        pid_t pid, pid1;
        /* fork a child process */
        pid = fork();
        if (pid < 0) { /* error occurred */
                fprintf(stderr, "Fork Failed");
                return 1;
        }
        else if (pid == 0) { /* child process */
                pid1 = getpid();
                printf("child: pid = \%d",pid); /* A */
                printf("child: pid1 = \%d",pid1); /* B */
        }
        else { /* parent process */
                pid1 = getpid();
                printf("parent: pid = \%d",pid); /* C */
                printf("parent: pid1 = \%d",pid1); /* D */
                wait(NULL);
        }
        return 0;
}
```

Question 8 (2 points): Including the original process, how many processes are created by the program shown below?

```
#include <stdio.h>
#include <unistd.h>
int main(){
        int i;
        for (i = 0; i < 5; i++)
                fork();
        return 0;
}
```