

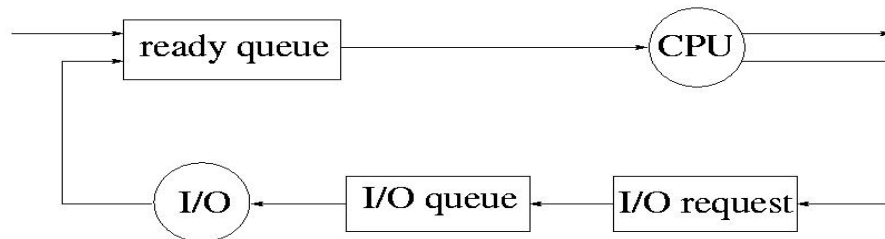
## CS 201 OPERATING SYSTEMS (FALL 2005)

ASSIGNMENT 2 (DUE Oct 20, before class. How to turn in the programs will be announced separately before the due date.)

1. Do the following exercises from the text book: 3.10, 3.13, 4.1, 5.1 (no programming is necessary; just describe), 5.6, 6.1, 6.3, 6.5

For Questions 2.

- Prime number is a positive integer that has no divisor other than 1 and itself. 1 is not a prime number.
  - To determine if  $n$  is prime try all possible divisors from 2 to the square root of  $n$ .
  - When asked a list of primes output them in ascending order.
2. Write two C programs to run in UNIX. One of the programs should take as a command level argument a positive integer, check if it is prime, and report the result. The other program should take a command-level integer argument, use a fork statement to create a child process that executes the other program with the same argument The parent process should report the completion of the child process (Hint: Similar to the example in Figure 4.8 in the textbook).
  3. Write a program(s) (C in UNIX) which simulates the events in a multiprogrammed computer system according to the following guidelines:
    - The scheduling is non-preemptive. Executing process is allocated the CPU until the end of its current CPU burst. The following queueing diagram describes the process scheduling:



- There are 5 I/O devices in the system. Implement a separate queue for each device.
- You may assume that the number of processes does not exceed 10.
- Use a counter to implement a logical clock.
- At each increment of the counter your program should check if the current counter value (time) matches with an arrival time of any process, and enter arriving process(es) automatically into the ready queue.
- An input file includes information for all processes, their arrival times, and sequence of CPU and I/O bursts. Input lines are of four possible formats:

```
BEGIN      <process number (<=10)>   <arrival time>
CPU        <CPU burst>
IO         <I/O burst>   <I/O device number (<=5)>
END /* Process description ends */
```

The following is an example:

```
BEGIN      1      0
CPU        3
IO         4      2
CPU        2
IO         3      4
END
```

```
BEGIN      2      5
CPU        5
IO         3      4
END
```

- You should not change the input file format as we will test your program with our input files.
- You may assume that the input is correct.

Your program should report which process executes, and which process performs I/O on which device, as there occurs a change. For example see the output below for the input given above.

- Report the events in the following format:

```
TIME: 0   PRO 1 EXEC
TIME: 3   PRO 1 I/O ON DEV 2
TIME: 5   PRO 2 EXEC
TIME: 10  PRO 1 EXEC
TIME: 10  PRO 2 I/O ON DEV 4
TIME: 13  PRO 1 I/O ON DEV 4
```