Midterm

CIS 415: Operating Systems

(80 minutes)

NAME: ______________________________________

Section	Total Points	Points Scored
1. Pthreads and Synchronization	20	
2. CPU Scheduling	20	
3. Disk Scheduling	20	
Total	60	

Comments:

1. Take a deep breath.
2. Write your name and initials on this page NOW and initial all other pages when you are told to start the exam.
3. Do all problems. Read each question carefully to be sure you are answering the question being posed.
4. Questions are intended to be answered succinctly. Concise and direct is better than rambling.
5. If you have a question, raise your hand and we will come to you, if we can. Otherwise, we will acknowledge you and you can come to us.

1. Pthreads and Synchronization

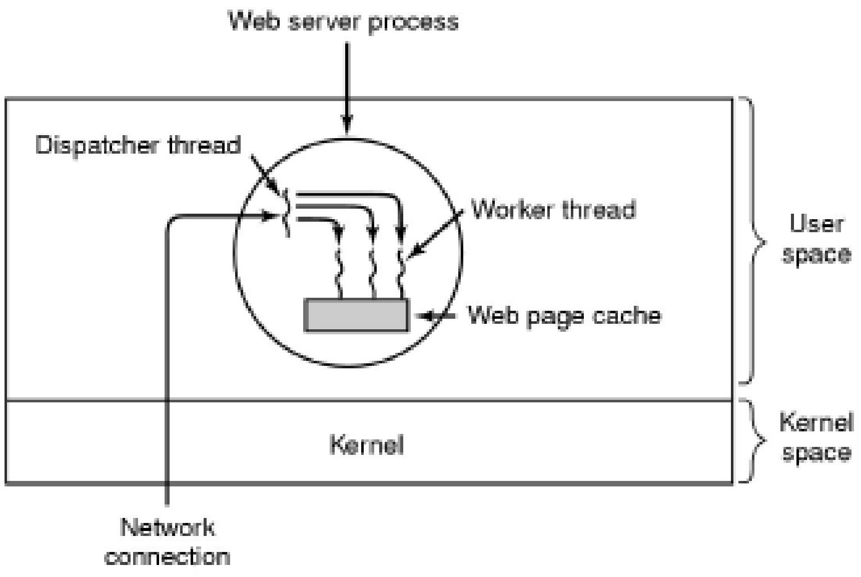
a) How does PThreads support the ability for threads to create critical regions around shared data? How does it support the creation of conditional critical regions around shared data? [6]

PThreads provides the abstraction of a mutex for creating critical regions; a mutex is used to achieve mutually exclusive access to shared data; each thread that wants to access data protected by a mutex must perform a lock(mutex)/unlock(mutex) sandwich around code that accesses the data.

To support conditional critical regions, PThreads provides the additional abstraction of a condition variable; a condition variable, when used together with an associated mutex, enables one thread to signal that an event has occurred that might be of interest to other threads; enables one to implement conditional critical regions based upon values of the shared state. after locking the associated mutex, code can test the shared state, and wait on the condition variable if it is not in the correct state to proceed.

For each type of critical region, 1 mark for name of mechanism, 2 marks for adequate description of abstraction and use.

b) One of the examples we discussed in lecture was a multi-threaded web server. In such an application, shown below, a dispatcher thread receives HTTP requests from the network, placing those requests in a queue; each worker thread obtains a request from the queue, reads the page from the disk, and returns the page to the requesting browser.



A multithreaded web server

Pseudo-code for the dispatcher function and the worker function are shown below. The queue behind the add_to_queue() and remove_from_queue() is unbounded; it is also ***not*** thread-safe; finally, remove_from_queue() returns NULL if there is nothing in the queue.

Since the queue is not thread-safe, the code as written is unsafe, since the dispatcher and worker threads may simultaneously attempt to modify the queue; additionally, the worker thread busy waits for a request.

Your initials _____

Sketch the changes needed to the code below to eliminate the unsafe race condition over the shared queue and to eliminate the busy wait loop in the worker function; do this using the PThreads support you have described in your answer to 1(a) above. [14]

Since it is unlikely that you remember the exact signatures for the PThread functions that provide the abstractions you described in 1(a) above, indicate immediately below the function signatures that you will be using in your modifications.

```
pthread_mutex_lock(&mutex);
pthread_mutex_unlock(&mutex);
pthread_cond_wait(&cond, &lock);
pthread_cond_signal(&cond);
```

(sketch your changes on the code below)

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;


void *dispatcher(void *args) {
   struct request *buf;
   while(1) {
      get_next_request(&buf); /* thread blocks waiting for network request */
      pthread_mutex_lock(&mutex);
      add_to_queue(&buf);      /* append to queue */
      pthread_cond_signal(&cond);
      pthread_mutex_unlock(&mutex);
   }
   return NULL;
}

void *worker(void *args) {
   struct request *buf;
   struct page *pg;

   while(1) {
      pthread_mutex_lock(&mutex);
      while ((buf = remove_from_queue()) == NULL) /* remove request from queue */
         pthread_cond_wait(&cond, &mutex);
      pthread_mutex_unlock(&mutex);
      if (! obtain_page_from_cache(&buf, &pg)) {
         read_page_from_disk(&buf, &pg);
         add_page_to_cache(&buf, &pg);
      }
      send_page(&buf, &pg); /* return page to requester */
   }
}
```

Your initials _____

2 marks for declaration and initialization of mutex, 2 marks for declaration and initialization of cond variable, 2 marks for lock/unlock sandwich around add_to_queue(), 2 marks for signal() or broadcast() after add_to_queue(), 2 marks for lock/unlock sandwich around remove_from_queue(), 4 marks for while() cond_wait() to eliminate busy wait.

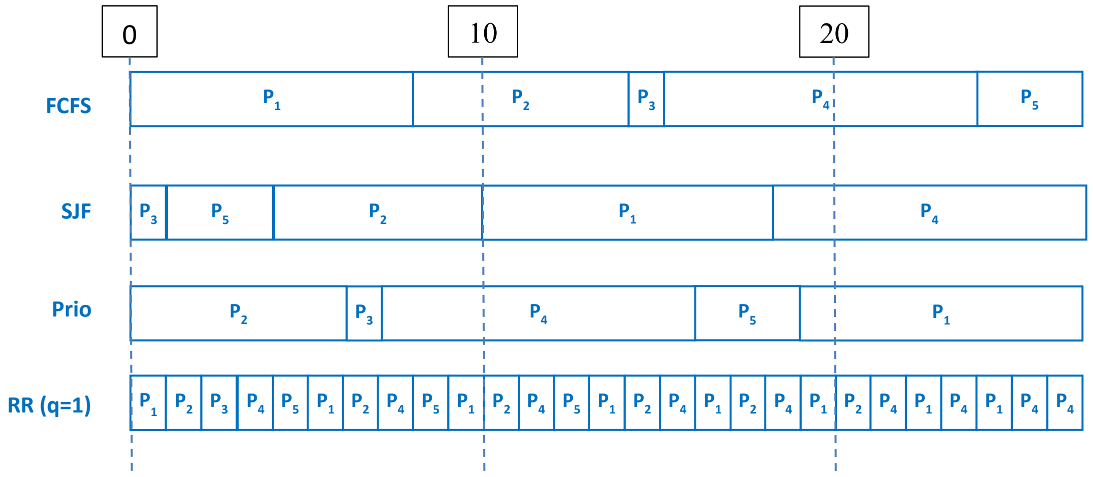
2. CPU Scheduling

a) Consider the following set of processes, with the CPU bursts given in milliseconds:

Process	Burst Time	Priority
P1	8	4
P2	6	1
P3	1	2
P4	9	2
P5	3	3

It is assumed below that all processes have arrived at time 0, and the initial ordering of the ready queue is P1, P2, P3, P4, P5.

(i) Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: First Come First Served, Shortest Job First, Non-Preemptive Priority (a smaller priority number implies a higher priority) and Round Robin (quantum = 1). [16]



Your initials _____

(ii) For each schedule above, compute the average wait time and average turnaround time, completing the table below. Which scheduling algorithm yields the minimum average wait time? Minimum average turnaround time? [4]

Algorithm	Average Wait Time	Average Turnaround Time
FCFS	12.2	17.6
SJF	6.6	12.0
NonP Priority	9.6	15.0
RR	12.4	17.8

FCFS:

- ave wait time = (0+8+14+15+24)/5 = 61/5 = 12.2
- ave turnaround time = (8+14+15+24+27)/5 = 88/5 = 17.6

SJF:

- wait time = (0+1+4+10+18)/5 = 33/5 = 6.6
- turnaround time = (1+4+10+18+27)/5 = 60/5 = 12.0

NonP Priority:

- wait time = (0+6+7+16+19)/5 = 48/5 = 9.6
- turnaround = (6+7+16+19+27)/5 = 75/5 = 15.0.0

RR:

- wait time:
  - P1 = 0+4+3+3+2+2+2+1 = 17
  - P2 = 1+4+3+3+2+2 = 15
  - P3 = 2
  - P4 = 3+3+3+3+2+2+1+1 = 18
  - P5 = 4+3+3 = 10
  - ave wait time = (17+15+2+18+10)/5 = 62/5 = 12.4
- turnaround time = (25+21+3+27+13)/5 = 89/5 = 17.8

Min average wait time: SJF

Min average turnaround time: SJF

Your initials _____

3. Disk scheduling

a) Describe the operation of each of the following disk-scheduling disciplines (be sure to state any additional information that is required by each discipline beyond the queue of pending requests):

(i) FCFS [2]

Simply service the disk requests in the order in which they arrive in the queue of pending requests. This is a fair discipline, but leads to significantly more disk head movement than most other scheduling disciplines.

(ii) SSTF [2]

Selects the request from the queue of pending requests that minimizes the movement of the disk head in an attempt service all requests that are close to the current head position. Needs to know the current head position at all times.

(iii) LOOK [2]

It continuously scans back and forth across the disk, servicing requests as it reaches the requested cylinder. Needs to know the current head position and current direction of travel, usually in terms of the cylinder numbers of the current request and the last serviced request. Reversed direction when there are no more requests in the queue in the current direction of travel.

b) SSTF is unfair (i.e. starvation can occur). Explain why this assertion is true. [2]

Since requests are arriving continuously, it is possible for particular requests in the queue to always be farther away from the current head position than other, more recently arrived requests.

c) Suppose that a disk drive has 2,000 cylinders, numbered 0 through 1,999. The drive is currently serving a request at cylinder 750, and the previous request was at cylinder 600. The queue of pending requests, in FIFO order, is:

100, 1475, 725, 1775, 950, 1500, 1025, 1750, 175

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests using the LOOK scheduling discipline? [5]

Current	Dir	Next	Difference
750	+	950	200
950	+	1025	75
1025	+	1475	450
1475	+	1500	25
1500	+	1750	250
1750	+	1775	25
1775	-	725	1050
725	-	175	550
175	-	100	75
		TOTAL	2700

1 mark for going in the right direction initially; 1 mark for turning around after completing request at 1775; 2 marks for total in range [2650-2750]; 1 mark for exactly 2700

Your initials _____

d) LOOK can lead to significant variance of the wait times for requests compared to FCFS.

(i) Indicate why this is so, and describe a way to modify the LOOK algorithm to reduce this variance. [2]

Significant variance due to alternation of direction. Can reduce the variance by Circular version of LOOK. Student must describe how their circular algorithm works.

(ii) Apply your modified algorithm to the request queue in (c) above. How does your algorithm compare with LOOK in terms of total number of cylinders visited? [5]

Current	Next	Difference
750	950	200
950	1025	75
1025	1475	450
1475	1500	25
1500	1750	250
1750	1775	25
1775	100	1675
100	175	75
175	725	550
	TOTAL	3325

1 mark for loopback from 1775 to 100; 2 marks for total in range [3275-3375]; 1 for exact number, 3325; 1 for indicating that C-LOOK is larger than LOOK for this queue of requests.

Your initials _____