Com S 352
Assignment 3
Due: September 22, 2017

4.8 (8 points) Which of the following components of program state are shared across threads in a multithreaded process?
a. Register values
b. Heap memory
c. Global variables
d. Stack memory

4.11 (5 points) Is it possible to have concurrency but not parallelism? Explain.

4.17 (10 points) The program shown in Figure 4.16 uses the Pthreads API. What would be the output from the program at LINE C and LINE P?

```
1. #include <pthread.h>
2. #include <stdio.h>
3. #include <types.h>
4. int value = 0;
5. void *runner(void *param); /* the thread */
6. int main(int argc, char *argv[])
7. {pid t pid;
8. pthread t tid;
9. pthread attr t attr;
10. pid = fork();
11. if (pid == 0) { /* child process */
12. pthread attr init(&attr);
13. pthread create(&tid,&attr,runner,NULL);
14. pthread join(tid,NULL);
15. printf("CHILD: value = %d",value); /* LINE C */
16. }else if (pid > 0) { /* parent process */
17. wait(NULL);
18. printf("PARENT: value = %d",value); /* LINE P */
19. }
20. }
21. void *runner(void *param) { value = 5;
22. pthread exit(0);
23. }
```

6.14 (15 points) Consider the exponential average formula used to predict the length of the next CPU burst. What are the implications of assigning the following values to the parameters used by the algorithm?

a. $\alpha = 0$ and $\tau 0 = 100$ milliseconds
b. $\alpha = 0.99$ and $\tau 0 = 10$ milliseconds

6.16 (30 points) Consider the following set of processes, with the length of the CPU burst given in milliseconds:

Process Burst Time Priority
*P*1 2 2
*P*2 1 1
*P*3 8 4
*P*4 4 2
*P*5 5 3
The processes are assumed to have arrived in the order *P*1, *P*2, *P*3, *P*4, *P*5, all at time 0.
a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
c. What is the waiting time of each process for each of these scheduling algorithms?
d. Which of the algorithms results in the minimum average waiting time (over all processes)?

6.19 (5 points) Which of the following scheduling algorithms could result in starvation?
a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority

6.23 (7 points) Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority.
When a process is waiting for the CPU (in the ready queue, but not running), its priority changes at a rate α. When it is running, its priority changes at a rate β. All processes are given a priority of 0 when they enter the ready queue. The parameters α and β can be set to give many different scheduling algorithms.
a. What is the algorithm that results from $\beta > \alpha > 0$?
b. What is the algorithm that results from $\alpha < \beta < 0$?

7.(20 points) Convert the following program to use threads.

The following restrictions apply:

1) One thread will print "hello ", one thread will print "world", and the main function will print the trailing "\n", using just pthread_create(), pthread_exit();, pthread_yield(), and pthread_join()..

Hints & Tips:

1) You must use a synchronization method to ensure that the "world" thread runs after the "hello" thread.

2) You must use a synchronization method to assure that the main thread does not execute until after the "world" thread.

```
/* Include Files      */

#include <stdio.h>

/* External References           */

extern void world( void );
extern void hello( void );

void main( int argc, char *argv[] ) {
      world();
      hello();
      printf( "\n" );

      }

/* world - print the "world" part.          */

void world( void ) {
      printf( "world" );

}

/* hello - print the "hello" part.          */

void hello( void ) {
      printf( "hello " );

}
```