Operating System Labs July-Dec-2017

Assignment 3

Exercise 1: Write a multithreaded program that creates two threads. Thread one will display 25 even numbers and thread two will display 25 odd numbers. The output of the thread should be synchronized so that the output should be sequential.

Exercise 2:

There are a couple of problems with the following program. First of all, the threads do not have a chance to complete, because the main program terminates without waiting for them. Second, the threads are not synchronized and therefore the text output is garbled. Your task is to add semaphores to this program to synchronize the threads.

```
#include <stdlib.h>
#include <stdio.h>

#include <unistd.h> /* defines _POSIX_THREADS if pthreads are available */
#ifdef _POSIX_THREADS
# include <pthread.h>
#endif

#include <semaphore.h>
void *text(void *arg);
int code[] = { 4, 6, 3, 1, 5, 0, 2 };

int main()
{
  int i;
  pthread_t tid[7];
  for (i = 0; i < 7; i++)
    pthread_create(&tid[i], NULL, text, (void*)&code[i]);
  return 0;
}

void *text(void *arg)
{
  int n = *(int*)arg;

  switch (n)
  {
    case 0:
      printf("A semaphore S is an integer-valued variable which can take only non-negative\n");
      printf("values. Exactly two operations are defined on a semaphore:\n\n");
      break;

    case 1:
      printf("Signal(S): If there are processes that have been suspended on this semaphore,\n");
```

```
      printf(" wake one of them, else S := S+1.\n\n");
      break;

    case 2:
      printf("Wait(S): If S>0 then S:=S-1, else suspend the execution of this process.\n");
      printf(" The process is said to be suspended on the semaphore S.\n\n");
      break;

    case 3:
      printf("The semaphore has the following properties:\n\n");
      break;

    case 4:
      printf("1. Signal(S) and Wait(S) are atomic instructions. In particular, no\n");
      printf(" instructions can be interleaved between the test that S>0 and the\n");
      printf(" decrement of S or the suspension of the calling process.\n\n");
      break;

    case 5:
      printf("2. A semaphore must be given an non-negative initial value.\n\n");
      break;

    case 6:
      printf("3. The Signal(S) operation must waken one of the suspended processes. The\n");
      printf(" definition does not specify which process will be awakened.\n\n");
      break;
  }

  pthread_exit(0);
}
```