ECEC 353: Systems Programming
Programming Exercise: Sleeping Barber

Prof. Naga Kandasamy
ECE Department
Drexel University

February 7, 2020

This programming exercise is worth 10 points. It is due via BBLearn by February 23, 2020, 11:59 pm. You may work on the project in a team of up to two people. If you are submitting as a group, please see the submission instructions later on in the document. You may discuss high-level concepts with other students but must write your own code.

Implement a solution for the *sleeping barber problem* using POSIX semaphores to coordinate the barber and the customers. The barbershop consists of a waiting room with n chairs and a barber room with one chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, then the customer sits in one of the free chairs and waits for his or her turn. If the barber is asleep, the customer wakes up the barber.

You have been provided with the following three files:

- *launcher.c* which launches the barber and customer processes, as shown by the following code snippet. The *rand_int()* function returns a random integer uniformly distributed within the provided interval, inclusive of the end points.

```c
#define MIN_CUSTOMERS 10
#define MAX_CUSTOMERS 20
#define WAITING_ROOM_SIZE 8

/* Create the barber process. */
switch (()pid = fork ())) {
    case 0: /* Child process */
        /* Pass the waiting room size as argument */
        snprintf (arg, sizeof (arg), "%d", WAITING_ROOM_SIZE);
        execlp ("./barber", "barber", arg, (char *) NULL);
    default:
        break;
}

/* Create the customer processes */
num_customers = rand_int (MIN_CUSTOMERS, MAX_CUSTOMERS);
for (i = 0; i < num_customers; i++) {
    switch (()pid = fork ())) {
        case 0: /* Child code */
            snprintf (arg, sizeof (arg), "%d", i);
            execlp ("./customer", "customer", arg, (char *) NULL);
        default:
            break;
}

/* Wait for the barber and customer processes to finish */
for (i = 0; i < (num_customers + 1); i++)
    wait (NULL);
```

- *barber.c* which contains the skeleton code for the barber. Please complete the program to achieve the desired functionality.
- *customer.c* which contains the skeleton code for the customer. Please complete the program to achieve the desired functionality.

- *simple_alarm.c* which contains code to help you implement the timeout mechanism within the barber program.

Please note the following when developing and testing your code:

- When naming your semaphores, make them unique with respect to semaphore names that other students might use. For example, if creating a semaphore for the barber chair, name it as */sem_barber_char_abc123*, where *abc123* is your alphanumeric Drexel ID.
- In the event your program deadlocks — which can be detected quickly as no progress will ensue between the barber and customer processes — manually terminate the three processes (using the *kill()* command). Then delete your semaphores that will be housed under `/dev/shm`. Otherwise, you will clog up the machine with a large number of running processes, making it unusable for other students.

Submitting Your Program

Once you have implemented all of the required features described in this document submit your code by doing the following:

- If using a make file, run `make clean` in your source directory. We must be able to built your programs from source and don't want your precompiled executables of intermediate object files. If your code does not at the very least compile, you will receive a zero.
- Zip up your code, including complete instructions in a separate README file on how to compile and run your programs on the *xunil* cluster.
- Name the zip file *abc123_chat.zip*, where *abc123* is your Drexel ID.
- Upload your zip file using the BBLearn submission link found on the course website.
- If you are working as a group of two, one submission will suffice. Please indicate the two authors as part of the header block within the *launcher.c* program.