Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO20-320202
Date: 2017-10-11
Due: 2017-10-18

OS Problem Sheet #3

Problem 3.1: *detectives meeting clients* (1+9 = 10 points)

A certain bar is a well-known hangout for detectives. If a detective comes to the bar and there are no clients at the bar, the detective talks to the bartender. If one or more clients are present, the detective leaves the bar together with all clients. If a client arrives and there are no detectives at the bar, the client orders a drink and waits. If there are one or more detectives, the client picks the detective who entered the bar first and leaves the bar together with the chosen detective.

Your task is to solve this synchronization problem using (counting) semaphors. Both, clients and detectives, are implemented as separate threads that execute a mainloop that looks like this (pseudocode):

```
enjoy_life(person)
{
    while (1) {
        switch (person->type) {
          case client:
            client_visit_bar(bar, person); break;
          case detective:
            detective_visit_bar(bar, person); break;
        }
        /* not very random but good enough here */
        usleep(random()%100000);
    }
}
```

a) Write a solution of the synchronization problem in pseudocode.

b) Implement the solution in C using POSIX semaphores and no other synchronization primitives (e.g., mutexes, condition variables, barriers, message queues, ...).

   Your program must support the command line option `-c` to define the number of clients (with a default value of 1) and the command like option `-d` to define the number of detectives (with a default value of 1).

   Your program should produce a trace of what is happening in the bar. The following trace consists of trace lines where each trace line consists of a column showing the name of the program, a column showing the number of clients in the bar, a column showing the number of detectives in the bar, and a column describing the event that caused the trace line to be generated.

   The trace below shows how client c0 enters the empty bar and decides to wait. Subsequently, detective d1 enters the bar, picks up client c0, and leaves the bar. The client c0 wakes up and leaves the bar as well.

```
$ ./bar -c 1 -d 1
bar:    0 c    0 d    c0 entering
bar:    1 c    0 d    c0 waiting...
bar:    1 c    0 d    d1 entering
bar:    1 c    1 d    d1 picking client c0
bar:    0 c    0 d    d1 leaving
bar:    0 c    0 d    ...c0 waking up
bar:    0 c    0 d    c0 leaving
```