Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: CO20-320202
Date: 2016-09-27
Deadline: 2016-10-04

Problem Sheet #2

Problem 2.1: *pthread programming* (1+1+1+1 = 4 points)

The following pthread source code has been written by a programmer who is less experienced than you. Help him to find and fix the mistakes he has made. Note that the program is syntactically correct, it compiles and links without any errors. Error handling code has been omitted for brevity, i.e., lack of runtime error checking is not the problem this programmer is facing. For each bug you have found, explain what is wrong with the code and indicate how the problem can be fixed.

```
/*
 * p2-pthread-oops.c --
 */

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

static int N = 5;

static void* run(void *arg)
{
    int *i = (int *) arg;
    char buf[123];

    snprintf(buf, sizeof(buf), "thread %d", *i);
    return buf;
}

int main(int argc, char *argv[])
{
    int i;
    pthread_t *pt = NULL;

    for (i = 0; i < N; i++) {
        pthread_create(pt, NULL, run, &i);
    }

    return EXIT_SUCCESS;
}
```

Problem 2.2: *student running group* (6 points)

After an intense weekend, a group of students decide to do something for their fitness and their health. They form a running group that meets regularly to go for a run together. Unfortunately, the students are very busy and hence not always on time for the running sessions. So they decide that whoever arrives first for a running session becomes the leader of the running session. The running session leader waits for a fixed time for other runners to arrive. Once the time has passed, all runners that have arrived start running. Since runners may run at different pace, not all runners complete the run together. But the runners show real team spirit and they always wait until every runner participating in the running session has arrived before they leave to study again.

In order to increase the pressure to attend running sessions, the students decide that whoever has missed five running sessions has to leave the running group. In addition, it occasionally happens that the leader of a running session has to run alone if nobody else shows up. This is, of course, a bit annoying and hence runners who did run ten times alone leave the running group voluntarily.

Write a C program to simulate the running group. Every runner should be implemented as a separate thread. The time between running sessions is modeled by xxxx. The time the session leader waits for additional runners to arrive is yyyy. Runners that arrive too late simply try to make the next running session.

Your program should use pthread mutexes and condition variables. Do not use any other synchronization primitives. Your program should implement the option `-n` to set the number of runners initially participating in the running group (default value is one runner).

While you generally have to handle all possible runtime errors, it is acceptable for this assignment to assume that calls to lock or unlock mutexes or calls to wait or signal condition variables generally succeed. (This rule hopefully keeps your code more readable and makes it easier to review your solutions.) In general, try to write clean and well organized code. Only submit your source code files, do not upload any object code files or executables or any other files that are of no value for us.

Below is the non-debugging output of a solution for this problem:

```
./runner -n 10 2>/dev/null
r0 stopped after 4 run(s). He missed 5 run(s) and did run 2 times alone.
r1 stopped after 0 run(s). He missed 5 run(s) and did run 0 times alone.
r2 stopped after 0 run(s). He missed 5 run(s) and did run 0 times alone.
r3 stopped after 1 run(s). He missed 5 run(s) and did run 1 times alone.
r4 stopped after 10 run(s). He missed 4 run(s) and did run 10 times alone.
r5 stopped after 4 run(s). He missed 5 run(s) and did run 4 times alone.
r6 stopped after 2 run(s). He missed 5 run(s) and did run 2 times alone.
r7 stopped after 3 run(s). He missed 5 run(s) and did run 2 times alone.
r8 stopped after 1 run(s). He missed 5 run(s) and did run 1 times alone.
r9 stopped after 11 run(s). He missed 4 run(s) and did run 10 times alone.
```