

RTOS, Spring 2015 – Lab #3: Processes and threads

Paolo Torroni, paolo.torroni@unibo.it

Davide Chiaravalli, davide.chiaravalli@studio.unibo.it

Objective: to learn the basics of programming with processes and threads using POSIX and Pthreads

1. Background

Make sure you have read and understood Chapters 3 and 4 from the OS textbook (“Process Concept” and “Multithreaded Programming”; in particular 4.4.1 “Pthreads”).

2. Hello World with Eclipse

A) Start up Eclipse. If prompted to select/create workspace, say OK

B) Create new “Hello World” project

(New → C Project → Hello World Ansi C Project).

Select Linux GCC compiler, give name to project (es, Lab1) then “Finish”

Note: you can install all this software on your computer at home. It's free. Check VirtualBox.org, Xubuntu.org, Eclipse.org

3. POSIX: Fork a process

A) Replace the existing code with that of the “sample fork program” below.

(The program can be downloaded from <http://lia.deis.unibo.it/Courses/RTOS/>)

Save (CTRL+S). Build all (CTRL+B) and execute (menu “Run”).

```
/**
 * Sample fork program
 */

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
int value = 5;
int main()
{
    pid_t pid;
    pid = fork();
    if (pid == 0) { /* child process */
        value += 15; printf("CHILD: value = %d\n",value); // LINE A
        return 0;
    }
    else if (pid > 0) { /* parent process */
        printf ("PARENT: value = %d\n",value); // LINE B
        wait(NULL); return 0;
    }
}
```

D) Use system call sleep(1) to introduce delays between instructions, compile again and check the output

E) Modify the program so that

- PARENT and CHILD print the same value, and
- CHILD prints before PARENT.

4. Pthreads: use the system calls to create and join

A) Create a new project with the “sample thread program” below (adapted from Silberschatz). (<http://lia.deis.unibo.it/Courses/RTOS/>)

B) Make sure you understand now the program works. Build all and execute.

```
/**
 * Sample thread program
 * [...]
 */

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

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* the thread */

int main(int argc, char *argv[])
{
    pthread_t tid; /* the thread identifier */
    pthread_attr_t attr; /* set of attributes for the thread */
    char arg[10]; sprintf(arg, "%d", 10); /* in the book, this value
                                           is taken from command line */

    /* get the default attributes */
    pthread_attr_init(&attr);

    /* create the thread */
    pthread_create(&tid, &attr, runner, arg);

    /* now wait for the thread to exit */
    pthread_join(tid, NULL);

    printf("sum = %d\n", sum);
}

/**
 * The thread will begin control in this function
 */
void *runner(void *param)
{
    int i, upper = atoi(param); sum = 0;
    if (upper > 0) {
        for (i = 1; i <= upper; i++)
            sum += i;
    }
    pthread_exit(0);
}
```

C) What is `atoi(param)`?

D) In general, a thread's function has one argument (for example, `runner` has `param`), which may or may not be `NULL`. When could that become useful?

Notice that, to correctly use Pthreads (with Eclipse), you should:

1) `#include <pthread.h>`

2) add the `-pthread` option both in the GCC C Compiler and in the GCC C Linker:

->Project->Properties->C/C++ Build->Settings->Tool Settings

->GCC C Compiler->Miscellaneous-> Other flags: add **-pthread**

->Project->Properties->C/C++ Build->Settings->Tool Settings

->GCC C Linker->Miscellaneous-> Linker flags: add **-pthread**

5. Turn a sequential computation into one that uses concurrent threads

A) Create a new project with the “sequential pi program” below.

(<http://lia.deis.unibo.it/Courses/RTOS/>)

```
/**
 * Sample pi program
 */

#include <stdio.h>
#include <stdlib.h>
#define N_STEPS (long int) 1000000

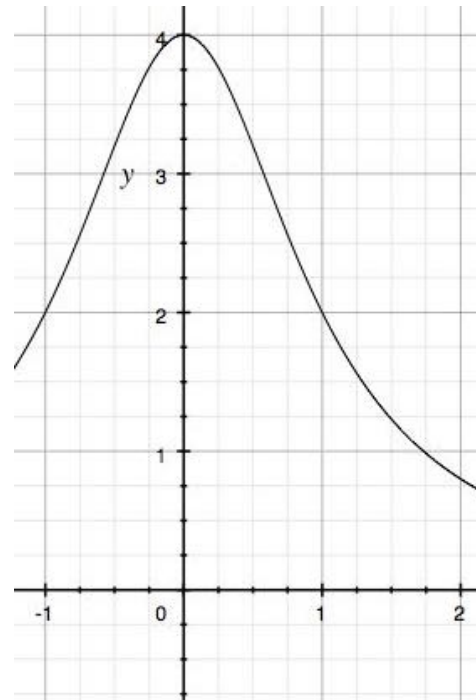
int main(void) {

    int i;
    double pi, step, x, sum=0.0;

    step = 1/(double)N_STEPS;

    for(i=0; i<N_STEPS; i++) {
        x = (i+0.5)*step;
        sum+=4.0/(1.0+x*x);
    }

    pi=step*sum;
    printf("Pi = %f\n", pi);
    return EXIT_SUCCESS;
}
```



B) Split the program into threads, so that the computation can be done in parallel by these threads. Use the Pthreads POSIX library.

Useful functions: `pthread_attr_init`, `pthread_create`, `pthread_join`, `pthread_exit`.
See **man** for information about these functions.

Note: if for some reason you wish to use the math libraries with Eclipse, you should:

1) `#include <math.h>`

2) add the `-lm` linking option in the GCC C Linker:

->Project->Properties->C/C++ Build->Settings->Tool Settings

->GCC C Linker->Libraries + **m**

`math.h` also defines a number of constants, for example π is `M_PI`

If you want to use these constant definitions, you should:

`#define _USE_MATH_DEFINES`

and then

`#include <math.h>`

6. Self assessment

- ☐ How can I produce an executable file out of a C program?
- ☐ I wrote a program in Eclipse. Now how can I execute it?
- ☐ What does the “sample fork program” print out at LINE A? and at LINE B?
- ☐ Is it possible to predict which line is printed out first (A or B)?
- ☐ What Pthreads function call is used to wait for a thread to finish?
- ☐ Does splitting the program into multiple threads make the execution faster?