
CS 466, Spring 2007
Assignment #2
The Great Race

Due 3/14/07, in class

Introduction

C (under Unix), Modula-3, Ada, and Java support programs with multiple processes. This assignment investigates the similarities and differences in the models used by these languages. To do this, you will write the same program in all four languages and then, after filling in some tables, identify and compare the models. (For those of you who are worried about not knowing Modula-3 or Ada, **DON'T PANIC**¹, I will provide you with most of the code.)

The program is a race between two processes. In the code, `stop`, `count`, `delay1`, `delay2`, and the iteration counters are (ACK) global variables and `STOP_AT` and `MAX_ITERATIONS` are constants. You may scale these constants (retaining the same ratio) to get a run time of about 20 seconds for the "100 100" run (see below). The need for scaling will depend on the hardware you are using.

Process 1

- (a) Print a welcome message
- (b) While `stop` is not true, `count` is less than `STOP_AT`, and the loop is iterated fewer than `MAX_ITERATIONS` times.
 - (i) Add one to `count`
 - (ii) Call the procedure `delay` passing `delay1`
- (c) When the loop in (b) stops, set `stop` to **true** and print the value of `count` and the number of iterations the loops for processes 1 and 2 have executed.

Process 2

- (a) Print a welcome message
- (b) While `stop` is not true, `count` is greater than `-STOP_AT`, and the loop is iterated fewer than `MAX_ITERATIONS` times.
 - (i) Subtract one from `count`
 - (ii) Call the procedure `delay` passing `delay2`
- (c) When the loop in (b) stops, set `stop` to **true** and print the value of `count` and the number of iterations the loops for processes 1 and 2 have executed.

When both processes have terminated, the main (parent) process should print the values of `stop`, `count`, the number of loop-iterations process 1 and 2 executed, and the value `count - iteration_count` for process 1 + `iteration_count` for process 2 (predict this value before you run the program).

Incomplete source code for `race.java`, `race.adb`, `race.m3` and `race.c` and a makefile that you **must** use are on the web page. The makefile includes the targets `c_race`, `m3_race`, `ada_race`, `java_race`, which will perform the actual races. For example, typing `make c_race` will run `race.c` twelve times (once for each line in the table) using the `time[1]` command.

¹ These are the best "large friendly letters" I could manage (see HHGtG).

What to Hand in (Part (d) will be a significant part of your grade)

- (a) A 2up listing of the source code (be sure and use `psf -2`).
`a2ps -q -Avirtual -2 -o mycode.ps <files>`
`lpr mycode.ps`
- (b) A 2up output of `script[1]` showing the 12 runs of each program (48 total runs).
- (c) The tables below, filled-in from the four races.
- (d) A short (2 to 3 page) write-up explaining the scripts and the numbers in the tables.
- (e) Finally, so I can run them, do the following:

```
foreach x (m3 c ada java)
  cp race_$x ~<your-account-name>/race_$x
end
```

Modula-3						
run	delay1	delay2	count	iteration count 1	iteration count 2	user time
1	100	100				
2	100	100				
3	100	100				
4	100	90				
5	100	90				
6	100	90				
7	90	100				
8	90	100				
9	90	100				
10	100	50				
11	100	50				
12	100	50				

C						
run	delay1	delay2	count	iteration count 1	iteration count 2	user time
1	100	100				
2	100	100				
3	100	100				
4	100	90				
5	100	90				
6	100	90				
7	90	100				
8	90	100				
9	90	100				
10	100	50				
11	100	50				
12	100	50				

Ada						
run	delay1	delay2	count	iteration count 1	iteration count 2	user time
1	100	100				
2	100	100				
3	100	100				
4	100	90				
5	100	90				
6	100	90				
7	90	100				
8	90	100				
9	90	100				
10	100	50				
11	100	50				
12	100	50				

Java						
run	delay1	delay2	count	iteration count 1	iteration count 2	user time
1	100	100				
2	100	100				
3	100	100				
4	100	90				
5	100	90				
6	100	90				
7	90	100				
8	90	100				
9	90	100				
10	100	50				
11	100	50				
12	100	50				