CS 213, Fall 2000
Homework Assignment H4: Writing concurrent programs
Assigned: Nov. 16, Due: Wed., Nov. 29, 11:59PM

November 15, 2000

Dave O'Hallaron (`droh@cs.cmu.edu`) is the lead person for this lab.

The purpose of this assignment is to help you become more familiar with different styles of concurrent programming and exceptional control flow.

Logistics

You must work alone on this assignment. All the files you need are in the directory:

```
/afs/cs.cmu.edu/academic/class/15213-f00/H4
```

Start by copying the file `H4.tar` from that directory to a (protected) directory in which you plan to do your work. Then give the command: `tar xvf H4.tar`. This will cause 8 files to be unpacked into the directory: `README`, `Makefile`, `ics.c`, `ics.h`, `tfgets-main.c`, `tfgets.h`, `tfgets-proc.c`, `tfgets-thread.c`. You will be modifying and handing in the `tfgets-proc.c` and `tfgets-thread.c` files. Enter your Andrew login ID at the top of each these two files. Do this right away so you don't forget.

Description

The standard library function

```
char *fgets(char *s, int size, FILE *stream);
```

reads in at most `size-1` characters from `stream`, stores them into the buffer pointed to by `s`, and returns `s`. If we were to call `fgets` with the `stdin` stream,

```
char *s = fgets(s, BUFSIZE, stdin);
```

then fgets would block (possibly forever) until we typed something and hit the <return> key.

Your task is to develop a timeout version of fgets called tfgets. If the user types an input string within 5 seconds, then tfgets behaves just like fgets, immediately returning a pointer to s. If 5 seconds elapse without any keyboard input, then tfgets times out and returns NULL to the caller.

You will develop two different solutions, in two different source files, one based on processes, and the other based on threads:

- tfgets-proc.c: This solution will implement tfgets using processes, signals, and nonlocal jumps.
- tfgets-thread.c: This solution will implement tfgets using threads.

Type make to compile your solutions and link each of them with a simple driver routine (tfgets-main.c) that calls tfgets and prints the result. The output is a pair of binaries called tfgets-proc and tfgets-thread. If your solutions are working properly, these binaries will immediately echo the input string if you type something within 5 seconds. Otherwise they will timeout by printing "BOOM!". We will evaluate the correctness of your solutions using the tfgets-main.c driver.

Rules

- Do all your work on the fish machines.
- Do not use the alarm function in any of your solutions.
- Reap all child processes and peer threads before returning to the caller.
- Do not mix signals and threads.

Evaluation

Each solution is worth 5 points, for a total of 10 points.

Hints and suggestions

- To help you get started, we have included templates for tfgets-proc.c and tfgets-thread.c. You may modify these files any way you wish.
- Expect your solution to be about 60–80 lines of C code. Our process-based solution uses the following Linux functions: fork, wait, signal, sleep, kill, sigsetjmp, siglongjmp, and pause. Our threads-based solution uses pthread_cond_signal and pthread_cond_timedwait, along with the usual calls for creating and reaping threads. We also use pthread_cancel, which kills a peer thread.

- Programming with processes, signals, and non-local jumps is described in detail in *Handout 6: Exceptional Control Flow*.
- Because your process-based solution will need to perform a nonlocal jump from a signal handler, you must use `sigsetjmp` and `siglongjmp`, rather than their `setjmp` and `longjmp` counterparts. Other than the fact that the former can be used from signal handlers, while the latter cannot, their behaviors and interfaces are identical.
- The `pause` function, which we haven't discussed in class, simply blocks the process and waits for a signal to arrive.
- We have included some error-handling wrapper routines in `ics.c` that might be helpful to you. You are not required to use them.
- Remember that `man` pages are your friends. For example, type `man pthread_cancel` to learn more about the `pthread_cancel` function.

Hand in

To handin your `tfgets-proc.c` and `tfgets-thread.c` files, type

```
make handin NAME=username
```

where `username` is your Andrew login ID. After the handin, if you discover a mistake and want to submit a revised copy, type

```
make handin NAME=username VERSION=2
```

You can verify your handin by looking in

```
/afs/cs.cmu.edu/academic/class/15213-f00/H4/handin
```

You have list and insert permissions in this directory, but no read or write permissions.