Washington State University Vancouver

Systems Programming - CS 360

Extra Credit - Due: 11:59PM Nov. 16th

Instructor:
Ben McCamish

Overall Assignment - Must get everything correct to get extra credit

Write a program (in C) called `extra.c` targeted at the Linux platform. Your program will spawn a child process with a pipe shared between the parent and child. The parent will write to the pipe and the child will read from it. There are no required command line arguments.

The **parent** process will intercept SIGINT and, when a SIGINT is detected by the parent, it will prompt the user for a single line text message. Once having received the message from the user, the parent will send (write) the message to the child via the shared pipe and send a SIGFPE signal to the child to get its attention. The parent process will repeat this behavior for any SIGINT it receives, until an "exit" message is received from the user (see below).

The **child** process must ignore SIGINT. When the child process detects a SIGFPE, it will read a single line message (less than 512 bytes) from the shared pipe. The first token of the message may optionally be an integer representing a time in seconds (DELAY). If the first token is not an integer, assume that the DELAY value is 5 seconds and the non-integer token is then part of the message. The child will print the message (minus the leading integer, if any) to standard output. It will repeat printing the message every DELAY seconds until the next SIGFPE is intercepted and a new message is read from the parent.

Below is an example where there is 10 seconds between each interrupt and input from the user.

Example:

```
# ./extra
^CInterrupt received, enter new message: 2 how are you
how are you
how are you
how are you
how are you
how are you
^CInterrupt received, enter new message: hows it going
hows it going
hows it going
```

Specifications and Restrictions

- (Required) Makefile containing at least 3 rules:
  1. `all:` (compiles everything together and produces an executable)
  2. `clean:` (removes all object and temporary files)
  3. `run:` (command for running your executable that works with the submitted code)
- (Required) Program must work on the lab machines with various forms of input. Consider edge cases carefully.
- (Required) Must be robust, including error catching. You must catch errors and print out an appropriate error message containing the `errno` and the message produced by that error. This means you will need `errno.h` and `string.h`, libraries at least.
- (Required) Implementation of the repetition interval shall be done using `signal()`, `pause()` and `alarm()`. Use of the library function `sleep()` and busy wait loops are not permitted.
- (Required) If the parent receives a message of "exit" from the user, the parent forwards the message to the child and waits for it to exit (terminate) and then performs its own exit. Upon receipt of the "exit" message, the child prints a message to standard out indicating its intention to exit and then exits.
- Helpful functions: `fork`, `signal`, `pause`, `alarm`, `close`, `exit`, `fgets`, `sscanf` etc. (consult man pages as needed)

What to turn in (in a zip on Canvas, yes Canvas):

- extra.c (no header files)
- Makefile