Assignments

Assignment 3 - fork and exec - Returned

Title	Assignment 3 - fork and exec
Student	Ryan Coslove
Submitted Date	
Grade	**11.00 (max 12.00)**

History

Instructions

This is an individual project. Submit only your source code. There is no need to create an archive file or include a makefile. If your code requires specific compiler options, note them in comments.

Update: 2021-03-25: Added argument for page width

Base Program [6 points]

We want a program that word-wraps several files and concatenates them. We have a working executable for `ww`, but no source code, so we cannot modify it for our purposes.

This almost works:

```
$ cat file1 file2 file3 | ./ww 80
```

Unfortunately, it may result in the last paragraph of one file being combined with the first paragraph of the next. (**Exercise**: Why is this?)

Instead, write a program `wcat` that takes a positive integer and one or more file names as arguments. For each file, it will use fork to spawn a child process, and the child process will use execl or execv to execute ww for a single file, using the provided integer as the page width. Ensure that a blank line separates the last paragraph of one file from the first paragraph of the next.

Note that the child process shares standard output with the parent process, so `ww` will print to the terminal without any work on your part.

Enhancement I [1 point]

Before invoking `ww`, make sure that the file name does not refer to a directory. If it does, print a message to standard error and skip the file. Return EXIT_FAILURE once all processing is complete.

Enhancement II [1 point]

Check the exit status of `ww` (that is, the exit status of the child process). If the status is not EXIT_SUCCESS, then `wcat` should return EXIT_FAILURE once all processing is complete.

Enhancement III [4 points]

Simply printing a newline between files may result in consecutive blank lines if one of the input files is empty or contains only whitespace. Use the technique described in class to arrange a pipe so that wcat can read the output of ww and copy it to standard output, but also detect whether there was any output at all.

You may assume that `ww` is well-behaved and will output either (a) at least one non-blank line, or (b) nothing, and that its output will not begin or end with a blank line.

Note 1

Use a macro to hard-code the location of ww. For example,

```
#ifndef WWPATH
#define WWPATH "/ilab/users/abc123/bin/ww"
#endif
```

When testing, it is fine to use something like "./ww" as the path as long as you only run wcat in the same directory as ww.

Note 2

When using `pipe`, be aware that open file descriptors are duplicated between the parent and child processes. In particular, the read end will not report EOF until both the parent and child have closed the write end.

Note 3

Recall that `dup2` lets us assign an open file to a specific file descriptor. For example, `dup2(x, 1)` will make standard output refer to the same file that x refers to.

Note 4

You may use `write` or `fprintf` to write to standard error. For write, use file descriptor 2. For `fprintf` or other `f*` functions use the global variable `stderr`.

Submitted Attachment

- wcat.c (1 KB; Apr 2, 2021 11:02 pm)

Additional instructor's comments about your submission

late policy

Back to list

- For Student, Faculty and Staff: Contact the Office of Information Technology: help@oit.rutgers.edu: 833.OIT.HELP

  Report an Accessibility Barrier

- Rutgers University
-