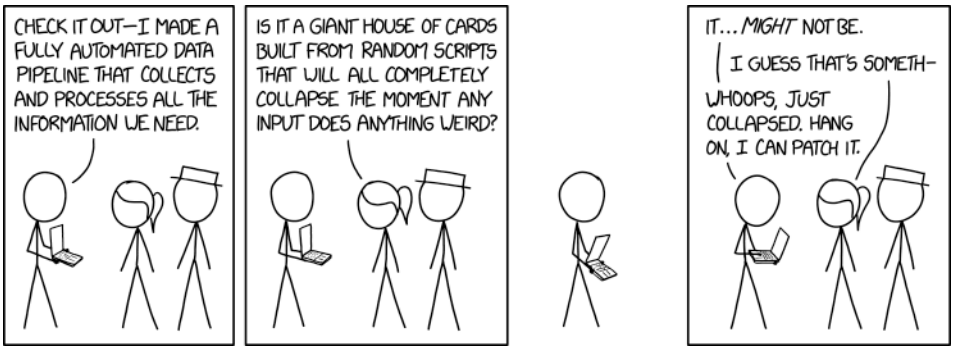
Program Assignment
CPE 357
Systems Programming

CAL POLY
SAN LUIS OBISPO

Assignment 4



Exercise:

Write a "findstuff" program.
The findstuff program will let you enter its own shell:
findstuff$
Now you should be able to enter some commands and react to them:
Important: I leave the exact formatting and text up to you, but hold up to the premise

- find <filename> -s .. tries to find this specific file in the current directory and all subdirectories (if –s is set). For that, your program fork()'s a new child process which is doing that. Assign a serial exclusive number to this and each child, you will need it later with the "kill" command.
  Once done, it interrupts* the scanf from the parent process and prints its result. The result should be something like >nothing found< in case nothing was found, but if, then print the file and its directory. Print several lines if several files with the same name were found.
  -s .. if this flag is set, search in all sub-directories. If not set, search only in the current directory. Also print the time in hh:mm:ss:ms format how long it took.

  E.g.
  findstuff$ find test.c //tries to find the file(s) named "test.c" in the current directory
  findstuff$ find text.txt –s //tries to find the file(s) named "text.txt" in the current directory and all subdirectories.

- find <"text"> -f:txt -s .. the behavior is similar to above, but here, the child tries to find a certain text in all files. For that, you need to open one file at a time, read the content into a malloc'ed memory and try to find the string. List all file and where to find them with this string.
  -f:XYZ .. means file ending. If this flag is set, XYZ (or more/less characters) representing the file ending. In that case search only in files with this ending.
  -s .. if this flag is set, search in all sub-directories. If not set, search only in the current directory.

  E.g.
  findstuff$ find "hello" //searches for the text "hello" in all files in the current folder.
  findstuff$ find "blabla" –f:c –s //searches for the text "blabla" in all c-files and in all subfolders.

- list .. lists all running child processes and what they try to do. Also displays their serial number.

- kill <num> kills a child process, and so ends its finding attemps.

- quit or q .. quits the program and all child processes immediatelly.

Have no more than 10 childs at a time. Report if the user attempts to exceed that limit and nicely print, that the limit is reached.

In a nutshell

Parent does the scanf. Childs are searching, reporting, and then ending.

*interrupt

How to interrupt while the parent is in scanf-mode waiting for another input?
Redirecting stdin! (see redirect video)
Make a pipe, redirect stdin, print and restore stdin.

What for?

Learning pipe(), dup2() and how to redirect stdin, lots of programing exercise due to string handling, nice use of signals and on top a useful program.

How we test

We send a couple of find request on the way on a big HD. So they will need some time (>3sec). Then we check if everything is nicely reporting back.

Submission:

Submit the source code file(s) and the executables: MYNAME_findstuff_ass4**.zip**

FAQ (this section will be extended in the next days)

>What happens with the child when its done? E.g. finding the file(s).

1. Report its findings by printing the result.
2. It should end. To avoid a zombie, wait for that specific child with waitpid() (https://linux.die.net/man/2/waitpid) which will be discussed in the Wednesday's video. How does the parent know to wait for that child? Signal! And send its PID into a pipe or shared mem!