Assignment 4 -- Final ush assignment

- $? processing and reporting signals
  - $? is exit value of the last command on which the shell waited

    or a specified value if the command did not exit

- Processing $? happens in several places:
  - When you wait on a process: (reporting signals happens here also)
    - determine if dead process called exit
    - set a global variable with exit value or error value
  - During built-in processing
    - built-in is successful -- set global value to 0
    - built-in is not successful -- set global value to 1
  - During expand() -- just turn global variable value into ASCII and add to expanded string.

Reporting Signals -- done after the waitXXX() system call returns:

- Determine if signaled
  - extract signal number, print signal text if not SIGINT
  - determine if core dumped, if so print " (core dumped)"

Command Expansion -- $(.....) processing

Done in expand:

- Find the $(
- Save the index of or pointer to the start of the command
- Find the matching)
- Temporarily store a end-of-string over the)
- Create a pipe (check for errors)
- Process the line with the write end of the pipe as stdout
  - processline() will need 2 new arguments, outfd, wait/don’t flag
  - processline() must not wait for the child to finish, just return
  - processline() should return the pid of the child started or an error value
- Close write end of pipe so next loop will get EOF
- Read LOOP
  - Best to directly read into the new expanded string
  - Keep reading until EOF or buffer full
- AFTER completing the read, if last character is \n "remove it", all other \n should be made spaces.
- Cleanup -- close read end of pipe, wait for child if one started
  - Remember the value for $?

Processline changes:

- Prototype: int processline (char *line, int outfd, int flags)
- Change to main, new call: processline(buffer, 1, WAIT);
  - output should go to stdout and we should wait for the child
- Change to processline:
  - in child only -- if outfd is not 1, put the outfd on 1
  - in parent only -- should we wait or not?
    - #define WAIT 1
    - #define NOWAIT 0
    - if flags say to wait, wait and process $? and report signals
    - if flags say to no wait, just return the pid of the child started

Processing order

- In main(): Remove Comments
- In processline():
  - Expansion
  - Pipeline identification
  - On each element of the pipeline / only element
    - Argument Parsing
    - Execution
      - Built-in or fork/exec

Implementing Pipelines

ps aux | grep xfce | grep -v grep | cut -c1-5

stdin -> ps aux -> stdout/pipe/stdin -> grep dh -> ...

processline()

- expand, then find pipelines
- "ps aux" a complete command
  - use processline ... but no expand, no wait
- loop over all pipeline elements ...
  - never need to have more than 2 pipes open at same time
  - ush needs to close both ends of every pipe opened
  - ush (parent and child) does not read or write to any pipes or files
- processline (line, infd, outfd, flags)

  flags => NOWAIT, NOEXPAND
- wait on last process in list (if this call waits)

  envset N $(ps aux | grep dh | grep -v grep | cut -c1-5)

  kill -9 ${N}