Project 2 – *The Cow Shell*

A mini command interpreter

due: 12:01 AM, Thursday, October 17, 2002

You may work individually or in groups of two or three on this project. Groups must be formed by Tuesday, October 8.

Introduction

For this project you will write a C program called the Cow Shell (`cowsh`), a mini command shell interpreter. `cowsh` is similar to familiar Unix shells such as the Bourne shell (`sh`) the Bourne-Again shell (`bash`), and C shell (`csh, tcsh`). You will learn about process creation, implementation of pipes, input/output redirection, background processes, signals, interrupt handling, and the use of some system calls.

Description

Like the familiar Unix shells, `cowsh` should issue a prompt (perhaps "`cowsh#`"), at which it reads commands from the user and executes them. When the user issues the `exit` command, `cowsh` should terminate.

Your shell should interpret the following commands:

- `exit`: exit from the shell
- `help`: display a message listing usage of all commands
- `run`: execute the command following it. That command may not be present in the current directory, in which case the directories in the `PATH` environment variable should be searched. Appropriate choice of `exec` function will help here. The arguments following the command should be passed to the command.

  For example,

  ```
  cowsh# run cat cat.c
  ```

  should execute `cat` with one argument, `cat.c`

  - *Input and output redirection* should be implemented.
    For example,

    ```
    cowsh# run cat < cat.c > myfile.c
    ```

    should cause the `cat` program to read from `cat.c` and write to the file `myfile.c`.
  - *Pipes* should be implemented.
    For example,

    ```
    cowsh# run cat cat.c | wc > count.txt
    ```

    should cause the output of `cat cat.c` to be the input of `wc > count.txt`.

  - <control-c> should abort the command being run, but **not** cause `cowsh` to terminate.

  You may wish to make the `run` part of this command optional, allowing programs to be executed by `cowsh` in a more familiar manner – by simply typing the command.

- `bg`: This is similar to `run`, except that it executes the command in the background.

  - Typing <control-c> should not kill commands running in the background.
  - When any background command terminates, it should be reported.

```
cowsh# bg sleep 55       ; invoke sleep in background
cowsh# run cat < hello.c ; give other commands
cowsh# ...               ; other commands
cowsh# ...               ; other commands
[2] "sleep" terminated   ; sleep command is done
```

  - All backgrounded processes should be maintained in a process table by `cowsh`, so that the program name is displayed when it terminates, and for use in the `jobs` and `kill` commands.

  You may wish to use the more traditional `&` at the end of a command to provide backgrounding of processes.

- `jobs`: Displays all the active background programs, along with their *id*s. (This id need not be that same as the actual process id, but it could be the index into your process table).

```
cowsh# jobs
PID     Name
[0]    mycat < myfile.c > newfile.c
[1]    idle 20
[4]    grep cowsh < doc | wc
```

- `kill`: Without any arguments, prints the usage:

```
cowsh# kill
kill <pid> [<pid> ...]
```

  Otherwise it kills the process with the specified ids and displays the process killed.

```
cowsh# kill 4
[4] "grep" Killed
cowsh# kill 3 7
[3] "cat" Killed
[7] "wc" Killed
```

- Errors should be reported meaningfully.
- Additional functionality of your group's choosing should also be implemented (see the Grading section below).

Implementation Notes

- The `system()` system call is **not** to be used.
- The system calls that you should use are `fork()`, a variant of `exec()`, `signal()`, `kill()`, `open()`, `dup2()`, `close()`, and `pipe()`.
- Use `cowsh` scripts to test your shell.
- The following might be a good order to tackle the required functionality.
  - `exit` and `help` commands
  - `run` command (with no argument passing, no redirection, no pipes)
  - `run` with argument passing
  - `run` with input and output redirection
  - `bg` command
  - `jobs` command
  - `kill` command
  - <control-c> trapping
  - trapping termination of background processes
  - pipes for `run`/`bg` commands
- `/home/faculty/terescoj/shared/cs432/project2/cowsh` contains my version of `cowsh`.

Grading

The project will be graded out of 75 points, and will be based on correctness, design, documentation, and style. An on-time submission that correctly implements and documents the required features can earn up to 65 points. The remaining 10 points are for the implementation and documentation of additional features of your choosing. Late submissions are not eligible for credit for additional features.

Submission

By 12:01 AM, Thursday, October 17, 2002, You should submit a file `cowsh.tar` that includes the following:

- Your fully commented C source file(s) and a `Makefile` to build the `cowsh` executable.

- Lucid implementation documentation. It should be such that a person who knows the problem but not your program should be able to follow your program easily and be able to change (or enhance) it without any problems (and not get bored reading it). This documentation should list your important design decisions, the assumptions that you made (if any), the additional functionality implemented, and any other relevant information.

Honor Code Guidelines

Collaboration within a group is unrestricted. You may ask the instructor or teaching assistant for help. Outside help (classmates, friends, reference manuals) with the programming language (syntax) or computer systems is permissible, but help with the design of your program is restricted to your group members, the instructor, and the teaching assistant. Outside references such as language manuals are permitted. Any other collaboration or consultation is prohibited and will be considered a violation of the Honor Code. If you wish to use or refer to any software libraries or outside source code beyond the standard C libraries, check with me first. If in doubt about anything related to Honor Code, ask now and avoid problems later!