Project 1
Wrap Up

Changing Directory

- #include <unistd.h>
  - int chdir(const char *path)
    - Changes working directory to passed in path
    - Returns 0 on success, -1 on failure
- Need to copy contents to $PWD to keep things consistent

cd path

- Convert relative path to absolute path
- Call chdir on absolute path
- Check return value
  - Print error message and exit if invalid
- Update $PWD using setenv
  - Need to use same absolute path as used in chdir

Getting the Time

- #include <sys/time.h>
  - int gettimeofday(struct timeval *tv, struct timezone *tz);
  - Stores current time into tv, current timezone in tz
  - Returns 0 on success, -1 on failure
- struct timeval
  - time_t tv_sec
  - suseconds_t tv_usec

etime cmd

- Record time1 with gettimeofday
- Fork child to execute cmd
- Waitpid on child to finish
- Record time2 with gettimeofday
- Output time2-time1
  - time2.tv_sec – time1.tv_sec
  - time2.tv_usec – time2.tv_usec
    - If negative, need to borrow from tv_sec

Sample README

Member 1: Exam Ple

Member 2: Rea D. Me

p1-Ple-Me.tar contents:

```
README
report.txt
main.c      // main implementation
main.h      // interface to impl., macros, etc.
util.c      // utility functions
util.h
Makefile
```

Sample README

Completed using: linprog

(alternatively)

Completed using: Ubuntu Linux 3.16.0-46-generic

(can use output of command 'uname -vr' here.)

To build:

$> make

To clean:

$> make clean

To run test suite:

$> make test

Sample README

Known bugs:

1. Hangs whenever my_parse is called.
2. Crashes whenever user doesn't input anything.
3. Crashes when it finishes running.
4. Does not catch zombies

To do:

1. Still need to do piping

Additional comments:

We used a static two dimensional array for the command sized at 100 elements with 256 bytes per element.

Sample Report

==========

Problem statement:

==========

(Either paraphrase the assignment goal or copy/paste the
assignment problem statement here.)

Implement a shell program.

==========

Steps taken to solve problem

==========

1. Experimented with existing shell.
2. Took notes on how it handled certain commands.
3. Wrote a few test programs to understand directory searching.
4. Implemented user-input routine.
5. Implemented input-parsing routine.
6. Implemented input redirection.
7. Implemented background process.
8. Experimented with our shell to make sure above were correct.
9. Implemented zombie-termination.
10. Implemented forking new process.
11. Completed.

Sample Report

==========

Assumptions

==========

No more than 255 characters would be used for input

Redirection and piping would not be mixed within a single command

==========

System calls

==========

read() - Get user input.

stdio.h - output, printf, etc.

stdlib.h - fork(), execv(), ...

...

Sample Report

==========

Problems

==========

- execv didn't seem to work as expected.
- Path searching was tricky.
- Weren't sure how much memory to allocate.
- Computer crashed once- lost some progress.

==========

Known Bugs

==========

(Just copy the details from the README.)

Sample Report

==========

Division of Labor

==========

Exam Ple

- user-input
- input-parsing
- process forking

Rea D. Me

- path-searching
- input redirection
- zombie termination

==========

Slack Days Used

==========

3 days for Exam Ple

2 days for Rea D. Me

Sample Report

==========
Log
==========
11/26/05
- project completed

11/25/05
- bug fixed: user-input > 120 characters crashes program
- speed up: path-searching improved.

11/24/05
- completed: input redirection
.
.
.
11/4/05
- design: some areas unclear.
- started implementation.

11/3/05
- researched existing shells.
- started designing.

11/2/05
- project assigned.

Sample Report

==========

Questions

==========

1. Is it safe to try to open a file that does not exist?

It can be unsafe if the program does not pay attention to the return value from the call to open(). If the program assumes that it has a pointer to a file that, and that file does not exist, it is possible that the program will crash before long, most likely with a segmentation fault.

Sample Report

==========

Additional Features

==========

Simple text auto-complete:

Pressing <Tab> will auto-complete the rest of the command argument to the first match. For commands, it completes to the first item in the $PATH. For arguments, it completes to first matching filename.