Project 3: The Shell

This project was written by instructors of CS354 at Purdue University, and is taken from their web site.

Introduction

The goal of this project is to build a shell interpreter like csh. The project has been divided in several parts. Some sources are being provided so you don't need to start from scratch.

Using the Debugger

It is important that you learn how to use a debugger to debug your C and C++ programs. If you spend a few hours learning how to use gdb, it will save you a lot of hours of development in this lab.

To start gdb type "*gdb program*". For example, to debug your shell type:

```
csh> gdb shell
```

Then type

```
(gdb) break main
```


This will make the debugger stop your program before main is called. In general, to set a breakpoint in a given function type "*break <function-name>*"

To start running your program type:

```
(gdb)run
```


Your program will start running and then will stop at *main*.

Use "*step*"or "*next*" to execute the following line in your program. "*step"* will execute the following line and if it is a function, it will step into it. "*next*" will execute the following line and if it a function it will execute the function.

```
(gdb) next      - Executes following line. If it is a function it will execute the function
and return.
or
(gdb) step      - Executes following line. If it is a function it will step into it.
```

An empty line in gdb will rerun the previous gdb command.

Other useful commands are:

```
print var    - Prints a variable
where        - Prints the stack trace
quit         - Exits gdb
```

For more complete tutorials on gdb see:

GDB Tutorial 1
GDB Tutorial 2
GDB Tutorial 3

GDB Tutorial 4

First part: Lex and Yacc

In this part you will build the scanner and parser for your shell.

- Download the tar file lab3-src.tar.Z, that contains all the files in lab3-src, to your home directory on CSSUN and untar it using the following command:
  ```
  uncompress lab3-src.tar.Z
  tar -xvf lab3-src.tar
  ```
- Build the shell program by typing :

  ```
  make
  ```

  To run it type:

  ```
  shell
  ```

  Then type commands like

  ```
  ls -al
  ```

  ```
  ls -al aaa bbb > out
  ```

  Check the output printed.

- Try to understand how the program works. First read the Makefile to learn how the program is built. The file command.h implements the data structure that represents a shell command. The struct *SimpleCommand* implements the list of arguments of a simple command. Usually a shell command can be represented by only one *SimpleCommand*. However, when pipes are used, a command will consist of more than one *SimpleCommand*. The struct *Command* represents a list of *SimpleCommand* structs. Other fields that the *Command* struct has are *_outFile*, *_inputFile*, and *_errFile* that represent input, output, and error redirection.

- Currently the shell program implements a very simple grammar:

  ```
  cmd [arg]* [> filename]
  ```

  You will have to modify shell.y to implement a more complex grammar

  ```
  cmd [arg]* [ | cmd [arg]* ]* [< filename] [ [> filename] [ >& filename] [>>
  filename] [>>& filename] ] [&]
  ```

- Insert the necessary actions in shell.y to fill in the *Command* struct. Make sure that the *Command* struct is printed correctly.

- Run your program against the following commands:

  ```
  ls
  ls -al
  ls -al aaa bbb cc
  ls -al aaa bbb cc > outfile
  ls | cat | grep
  ls | cat | grep > out < inp
  ls aaaa | grep cccc | grep jjjj ssss dfdffdf
  ls aaaa | grep cccc | grep jjjj ssss dfdffdf >& out < in
  httpd &
  ls aaaa | grep cccc | grep jjjj ssss dfdffdf >>& out < in
  ```

The deadline of this part of the project is March 18, 2003, before class. Follow these instructions to turnin your part one.

1. Login to CSSUN.

2. cd to lab3-src and type "make clean"

3. Type "make" to make sure that your shell is build correctly.

4. Type "make clean" again.

5. cd one directory above lab3-src by typing "cd .."

6. Create a tar file named <user_name>.tar, where <user_name> is your CSSUN login, by typing

    tar -cf <user_name>.tar lab3-src

7. Gzip the tar file by typing

    gzip <user_name>.tar

8. Since this timestamp will be used to verify whether the work was completed on time or not, you should set the permissions on the file you submitted to make sure that the file timestamp is not changed. So this by typing:

    chmod a-w <user_name>.tar.gz

9. Mail the gzipped tar file to clay at cs dot georgetown dot edu as an attachment.

Resources

Here are the man pages for Lex and Yacc.

Additional links to information about lex and yacc can be found here.