HPC Hardware Assignment 4

Michael Tierney 11348436

March 2016

Introduction

For this assignment one had to create a Linux shell using in built system calls. A shell is used to interact with the Linux kernel. It should be able to run commands, navigate directories, create background processes and handle IO rediects/ piping. The first step is parsing the the command.

Parsing

Parsing was carried out using c++ stringstreams. The command is separated based on three layers. The top layer is used to store separate process groups. These occur when ";" is encountered. The next layer is used to store commands that are all part of the same process group for example commands separated by "|". The final layer is used to separate the individual arguments present in each command. These separations are determined using a set of reserved characters ";,&,|,>,>>,<". After the commands have been parsed the second layer is permuted to allow for stack unwinding later. In this step IO redirection operators are moved to the right such that they are encountered first when unwinding the stack. The final step involves expanding wildcards, "∼" and other expressions using wordexp().

Changing Directory

Changing directory is done by performing chdir() if "cd" is encountered. Before the directory is changed the current directory is saved. cd "-" returns the user to this directory.

Command Execution

For each foreground process group detected the shell forks, passes terminal priority to the child and waits for the child to finish. The child process unwinds the command stack, forking and waiting at each step. Eventually the most derived child returns and this propagates up the stack. The shell detects this and continues to the next process group. If no more groups are detected then it prompts for more input using readline.

IO Redirection

If and IO redirection character is encountered specific events have to occur. For "|" pipe file descriptors are generated, the process forks and the corresponding input and output for the commands are connected to the two ends of the pipe using dup2. The parent then waits for the child to execute and the child continues unwinding the stack. For ">","1 >","2 >" and ">>","1 >>","2 >>" the process changes the output to the filename listed using open() and dup. Similarly "<" changes the input using open().

Background execution

When "&" is encountered the shell forks but does not wait for the child to finish and instead continues on. The pid of the child is put into an array and its state is checked before each new prompt. When the child process group finishes the shell will print that the process group has completed.