Mini Shell

Lab assignment

A shell is an interface between a user and the operating system. It lets us give commands to the system and start other programs. Your task is to program a simple shell similar to for example Bash, which probably is the command shell you normally use when you use a Unix/Linux system.

Preparations

When programming a shell, several of the POSIX system calls you studied already will be useful. Before you continue, make sure you have a basic understanding of at least the following system calls: fork(), execvp(), getpid(), getppid(), wait(), and pipe().

Parser

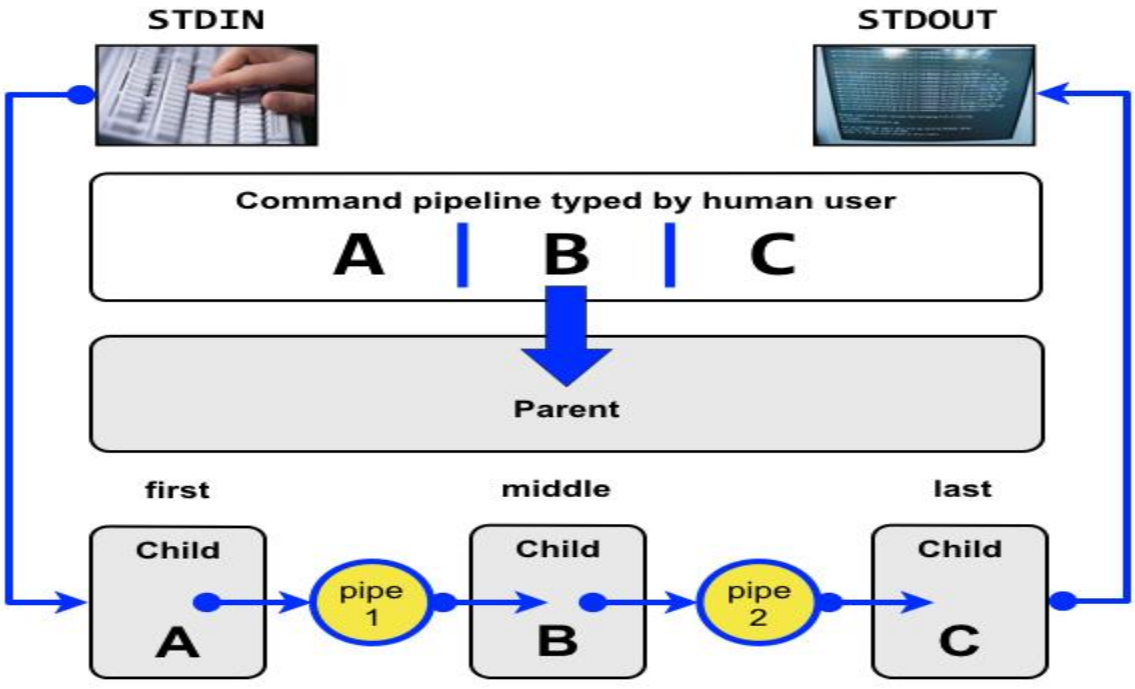
In slides/Lab_Assignment/src/parser.h you find the following prototype.

```
int parse(char *str, char* argvs[MAX_COMMANDS][MAX_ARGV]);
```

This function parse a command line string str such as "ls -l -F | nl" and populates argvs, an array with one argvvector for each command. The argv for the first command is stored at argvs[0], the argv for the second command in argvs[1] etc. The parse function returns the number of commands in the parsed pipeline.

Program design

The below figure shows the overall structure of shell.



When a user types a command line on the form A | B | C the parent parses the user input and creates one child process for each of the commands in the pipeline. The child processes communicates using pipes. Child A redirects stdout to the write end of pipe 1. Child B redirects stdin to the read end of pipe 1 and stdout to the write end of pipe 2. Child C redirects stdin to the read end of pipe 2.

Use slides/Lab_Assignment/src/shell.c to implement your solution. This file already implements the most basic functionality but it is far from complete.