# DESIGN DOCUMENT

## P1 bash-like shell

---

Problem Statement - Create a basic bash-like shell that supports the following features:

1. Can execute basic shell commands (ls, wc, cat etc.)
2. Supports 3 pipe operators - |, || and |||
3. Supports I/O redirection
4. Supports a new command - sc
5. Sending processes to the background using &

## Execution of a command

Consider the execution of the command: `ls -al | head | wc`

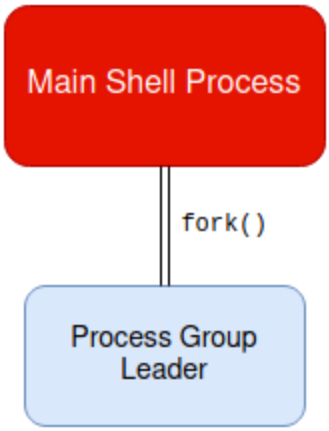
This single command consists of 3 sub-commands - ls, head and wc. The steps involved in the execution are as follows-

1. Shell takes user input and checks for 2 special commands right away. These are `exit` and the shortcut command `sc`. If it matches with either, they are executed within the shell process (without spawning child processes)

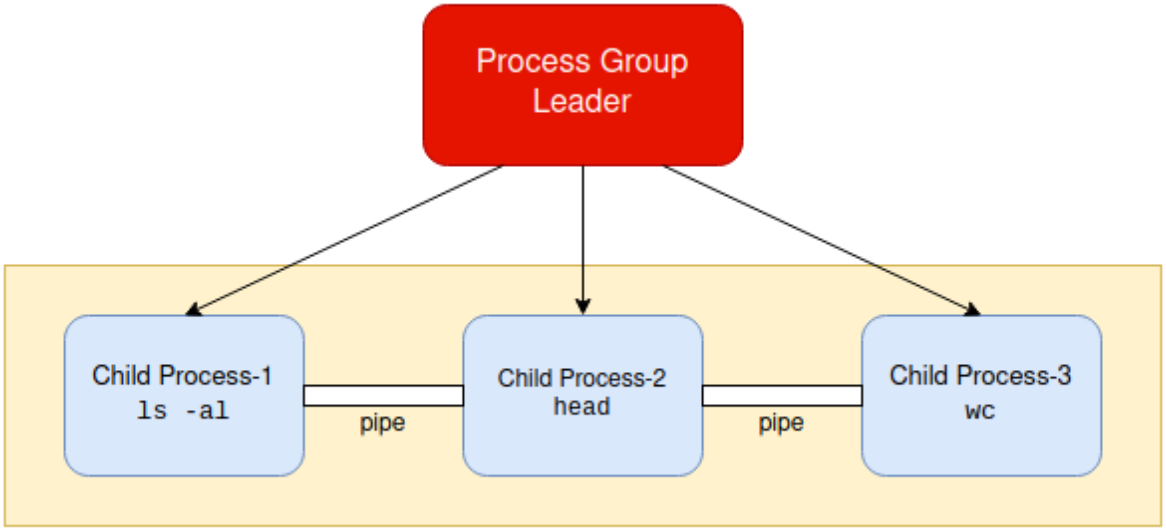
2. If no match is found(as in this case), the shell parses the command for pipe operators and passes the parsed command for execution.

3. A new process is forked. This process is made the leader of the process group in which the entire command will execute. Then the `tcsetpgrp()` call sends the main shell process to the background and brings the new process group to the foreground.

4. All sub-commands are executed within this process group by first searching for their canonical path in the directories listed in the `PATH` variable.




5. For each sub-command, a new child process is forked that executes it independently, while also handling I/O redirection. All the while, the parent waits for the child processes to finish or stop executing.
6. If the sub-commands are piped, a separate function handles the file descriptors using the `close()` and `dup2()` system calls.

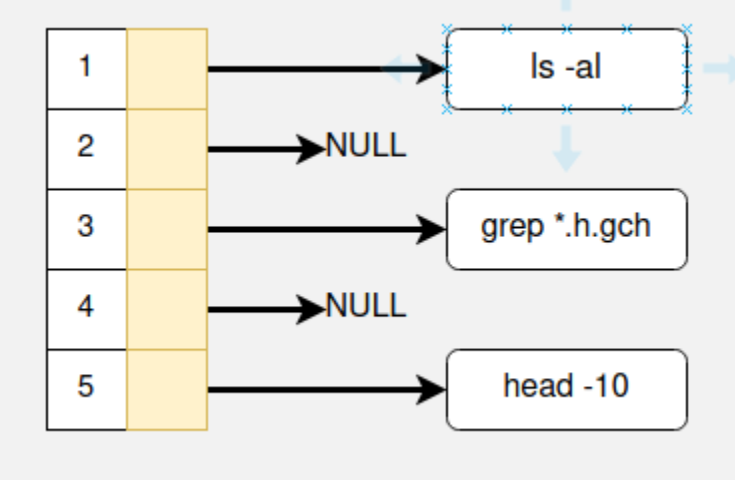



As in the figure, every sub-command's executing process communicates with its siblings via a pipe which handles the output of the writing process and pipes it to the input of the reading process.

## Executing the sc command

The shortcut command runs on a global data structure that resembles a lookup table. The table consists of integers mapped to commands. The user can enter the so-called 'shortcut mode' by pressing Ctrl C. Normally, this would terminate the current process but the signal generated by this keypress, `SIGINT`, has been handled in a signal handler, which sets a global variable `sc_mode`. This variable is checked before taking user input and if set, it takes an integer as input. The command (if any) mapped to this integer in the lookup table is then executed like any other command. Users can add and delete entries to the table using 2 commands:

- For insertion : `sc -i <index> <cmd>`
- For deletion : `sc -d <index>`




## Using pipe operators

The shell supports two new pipe operators - || and |||, that pipe the output of the first command to the next 2 and 3 commands respectively. These are mainly handled during the parsing phase. If a command contains these operators, it is split into separate commands that use the single pipe operator. So if the command is-

```
ls -al || sort, wc
```

Then the separated commands will be-

```
ls -al | sort
ls -al | wc
```

These are then executed sequentially. Similarly for the triple pipe operator.

## Usage

The shell can be compiled using the Makefile:

```
make
./shell
```