**Multicore Programming Project 1**

담당 교수 : 박성용

이름 : 한석기

학번 : 20211606

1. **개발 목표**

MyShell 프로젝트의 개발 목표는 System level에서의 Process control과 Process signal, Communication between process, 그리고 Background process의 개념을 Linux shell programming하면서 이해하는 것이다. 각 단계마다 fork(), exec(), pipe(), signal()을 이해하며 Programming한다.

1. **개발 범위 및 내용**
   1. **개발 범위**
2. Phase 1

-cd: 이동하고자 하는 directory로 이동한다.

-ls: directory의 file, directory의 리스트를 보여준다.

-mkdir: 새로운 directory를 생성한다.

-rmdir: 지우고자 하는 directory를 지운다.

-touch: 빈 파일을 생성한다.

-cat: 파일의 내용을 읽어준다.

-echo: 파일의 내용을 출력한다.

-exit: 모든 child process를 종료하고 shell을 종료한다.

-history: 현재까지 진행한 명령줄을 순서대로 출력한다.

-!!: 마지막으로 실행했던 명령줄을 실행한다.

-!#: #에 해당하는 라인을 history에서 찾아 실행한다.

1. Phase 2

-ls | grep filename: directory의 file, directory 중 filename이 포함된 리스트를 출력한다.

-cat filename | less: filename안의 내용을 페이지 별로 읽는다.

-cat filename | grep -i "abc | sort -r: filename안의 내용 중 "abc"가 대소문자 구분 없이 포함된 것을 내림차순으로 출력한다.

* 1. **개발 내용**
* **Phase1 (fork & signal)**

eval함수에서 process ID에 fork()함수를 이용해 child process를 생성한다. 만일 정상적으로 생성됐으면 process ID에는 0이 들어간다.

eval함수에서 child process가 생성되는데 parent process에서 waitpid를 사용해 child process가 종료될 때까지 기다린다. 그리고 parent process의 다음 코드를 이어서 실행한다. 만약 정상적으로 waitpid가 진행되지 않으면 waitpid는 음수를 출력한다.

* **Phase2 (pipelining)**

arguments를 읽어 pipe를 찾는다. pipe를 찾으며 개수를 count하고 배열에 pipe count를 넘버링해 인덱스로 활용하고 pipe index +1값을 value에 넣어줘 command의 위치를 저장한다. 이런 과정에서 pipe가 하나라도 존재할 시, flag에 1의 값을 주고 eval함수의 fork() 이후 과정에서 flag값이 1일 시, piping함수를 실행한다. 따라서 pipe가 하나도 없어도, 무수히 많아도 Pipe counting과 piping함수를 통해 정상적으로 실행한다.

piping함수는 Pipeline에 따라 앞 뒤 관계를 연결하고 해당 관계에 따라 명령어를 실행시키는 함수이다. 먼저 pipe의 개수대로 쓰기, 읽기를 저장할 pipe를 생성한다. 첫번째 명령어를 실행하고 파이프의 쓰기 끝을 표준 출력으로 복제한다. 이 때, dup2함수를 사용한다. 그리고 명령어를 실행하고 다음 파이프로 이동한다. 다음 파이프에서 표준 입력을 닫고 이전 파이프의 읽기 끝을 표준 입력으로 복제한다. 이 과정에서 이전 명령어의 출력이 다음 명령어의 입력으로 들어와 Pipeline의 역할을 수행한다.

* 1. **개발 방법**
* **Phase1 (fork & signal)**

main함수에서 append\_decision함수를 이용해 cmdline을 history에 추가할 것인지, 결정했다. append\_decision함수는 첫번째 char가 !가 아닐 때, history file을 읽고 history의 마지막 줄이 cmdline과 다를 때, history file에 추가하는 함수다.

eval함수에서 child process를 성공적으로 생성한 후, execve를 위한 경로인 "/bin/" + "argv[0]"을 추가한 char array를 생성하고. execve함수에 활용했다.

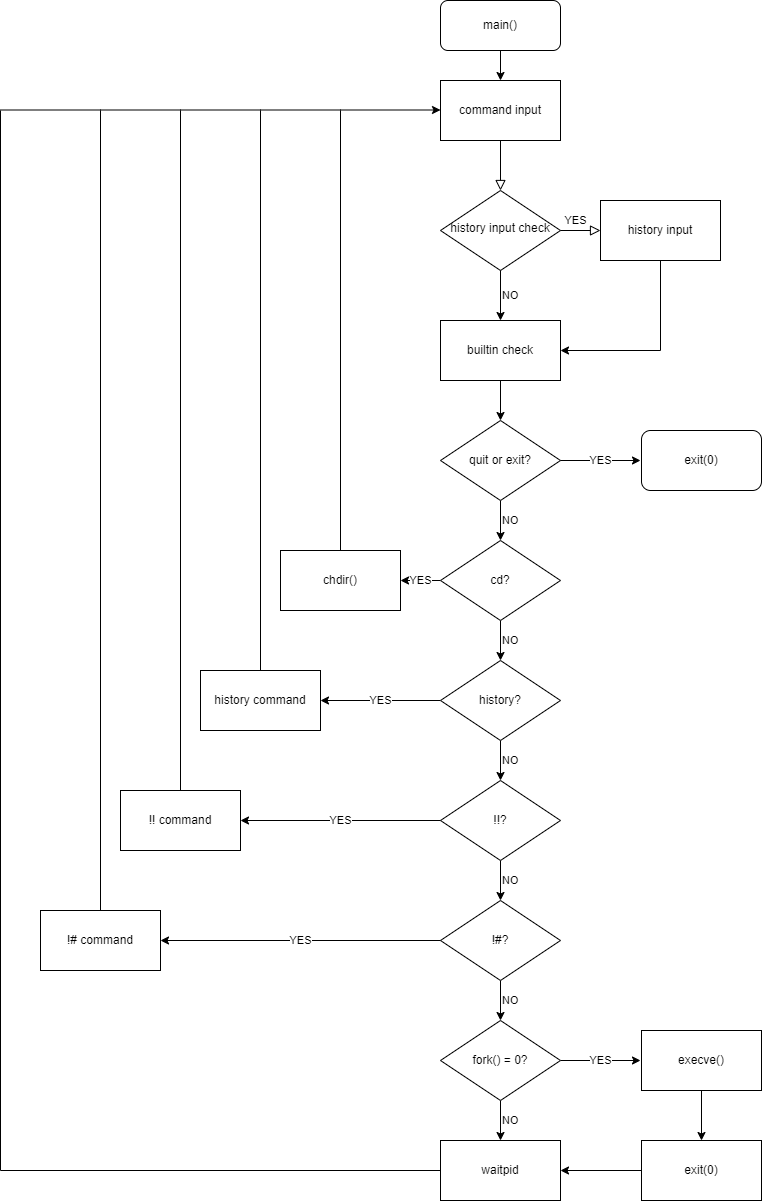
builtin\_command함수에서 "quit" command를 exit(0)으로 구현했다. "cd" command는 chdir함수를 이용해 구현했다. "history" command는 fopen의 "r"을 통해 history file을 읽으며 출력했다. !! command는 history file의 마지막 문장을 찾고 eval함수 인자로 넘겨 해결했다. !# command는 #이 숫자인지 확인하고 history file을 돌며 #에 해당하는 line을 append\_decision함수로 확인하고 eval함수 인자로 넘겨 해결했다.

* **Phase2 (pipelining)**

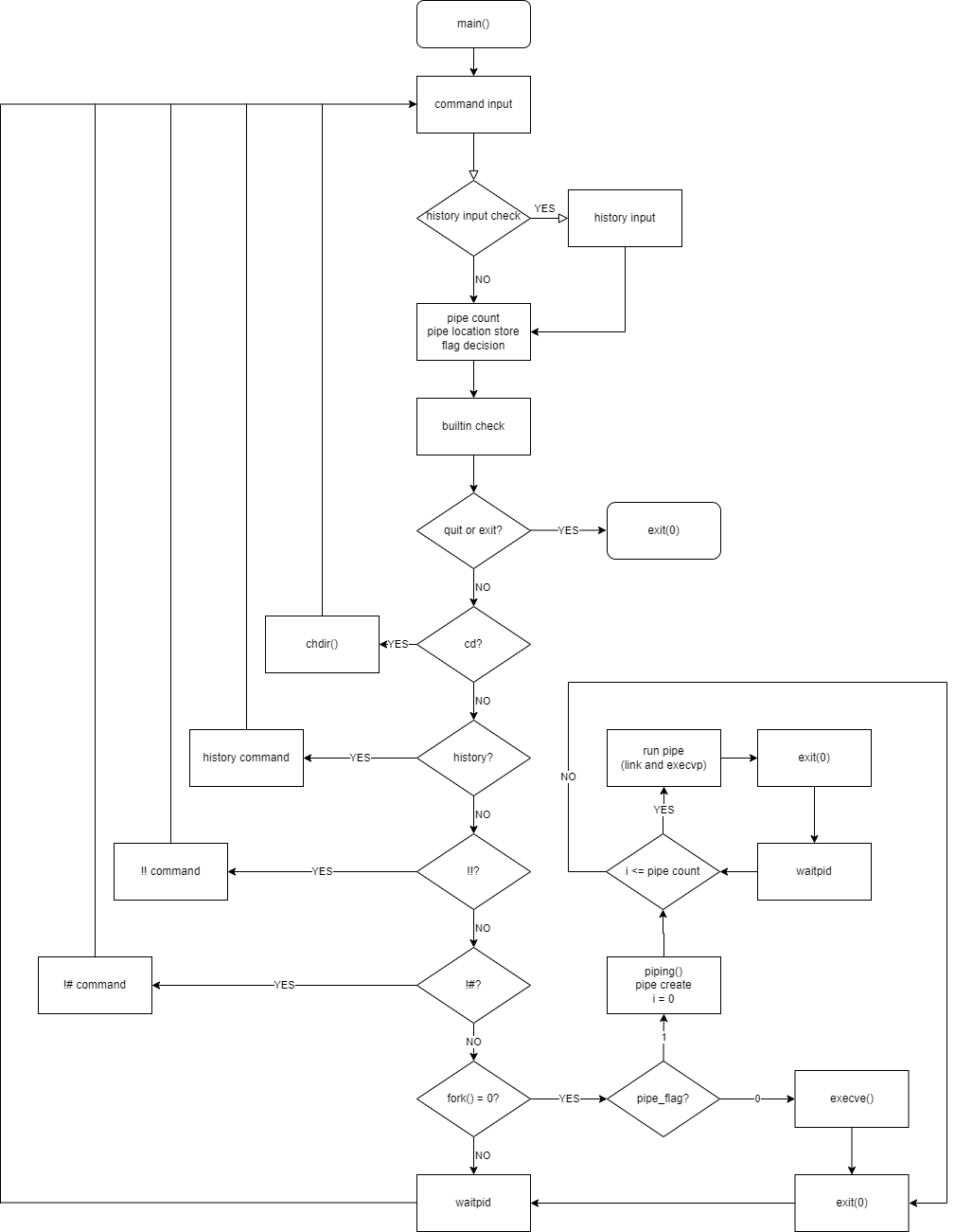
eval함수에서 pipe의 개수를 세고 위치를 저장한다. 그리고 pipe가 존재한다면 child process에서 piping함수를 실행한다.

piping함수는 pipe를 생성하고 pipe를 실행시키는 함수다. 먼저 모든 pipe를 생성한다. 그리고 pipe의 개수 + 1만큼 반복문을 돌며 pipe의 끝과 끝을 연결하는 작업을 실행한다. 해당 작업은 dup2()함수를 통해 실행했으며 pipe의 개수 +1인 이유는 명령어의 개수가 pipe의 개수 + 1이기 때문이다. 그래서 반복문의 처음이 아닐 때, 반복문의 마지막이 아닐 때를 구분했다. 그리고 parent process에서 명령어의 실행 횟수만큼 waitpid()함수를 실행해 child process가 종료될 때를 기다린 후 다음 명령을 실행한다.

1. **구현 결과**
   1. **Flow Chart**
2. **Phase 1 (fork)**

****

1. **Phase 2 (pipeline)**

****