**System Programming Project 1**

담당 교수 : 김영재

이름 : 최해민

학번 : 20181699

1. **개발 목표**

phase1,2,3를 순서대로 구현하면서 완성된 Shell을 만드는 것을 목표로 한다.

phase1 : 간단한 shell을 작성하는 것. 자식 프로세스와 부모 프로세스를 이용해서 기본 내부 단일 쉘 명령을 실행할 수 있어야한다. (cd, ls, mkdir, cat, echo, exit 등)

phase2 : fork()와 exec()을 이용해서 파이프 및 다중 파이프를 구현한다. 부모는 여러개의 자식을 호출하고, 부모와 자식들간에는 파이프로 통신이 가능해야한다.

phase3 : 백그라운드와 포그라운드에서 유동적으로 프로세스를 실행할수있어야한다.

1. **개발 범위 및 내용**
   1. **개발 범위**
2. Phase 1  
   cd, ls, mkdir, cat, echo, exit 등 기본 단일 명령이 제대로 시행되어야하며, 추가로 ctrl+c 명령을 내렸을때도 myshell은 여전히 구동된상태로 초기화가 되어야한다.
3. Phase 2

grep을 이용한 다중 파이프 명령이 무사히 수행되어야한다.

1. Phase 3

& 선언이 들어갔을때 백그라운드와 포그라운드에서 겹치지않게 프로세스가 잘돌아 가야한다.

* 1. **개발 내용**
* **Phase1 (fork & signal)**
  + fork를 통해서 child process를 생성하는 부분에 대해서 설명

명령 문자열을 받고, 해당 문자열이 exit명령이거나 따로 비어있지 않은경우에 child process를 pid=fork()를 통해서 시행시킨다. child process에서는 exec()를 프로세스가 돌아가며 exec()가 정상적으로 종료된 후에 child process도 종료되고, 부모 프로세스는 child process가 종료될때까지 대기하게 된다.

* + connection을 종료할 때 parent process에게 signal을 보내는 signal handling하는 방법 & flow

원래는 자식 프로세스가 종료가 되면 부모에게 SIGCHLD 신호가 전송되게 되어있고, 부모가 wait나 waitpid 함수로 이 자식프로세스의 상태정보를 받아서 자식프로세스가 정말로 종료가 되게 된다.

* **Phase2 (pipelining)**
  + Pipeline( ‘|’ )을 구현한 부분에 대해서 간략히 설명 (design & implementation)

먼저 argv를 탐색해 파이프가 있는지 확인한후, 파이브를 생성하는 함수를 호출한다. 그후에 파이프 변수는 2차원배열로 설정한 후, 파이프 개수만큼 사용하게 된다. 첫 파이프와 마지막 파이프는 각각 출력과 입력 부분을 close 해서 흘려주어야하고, 중간 파이프들은 두부분에 있어서 close를 해주어야 데이터가 흘려지기 때문에 세부분에 나눠서 구현한다.

* + Pipeline 개수에 따라 어떻게 handling했는지에 대한 설명

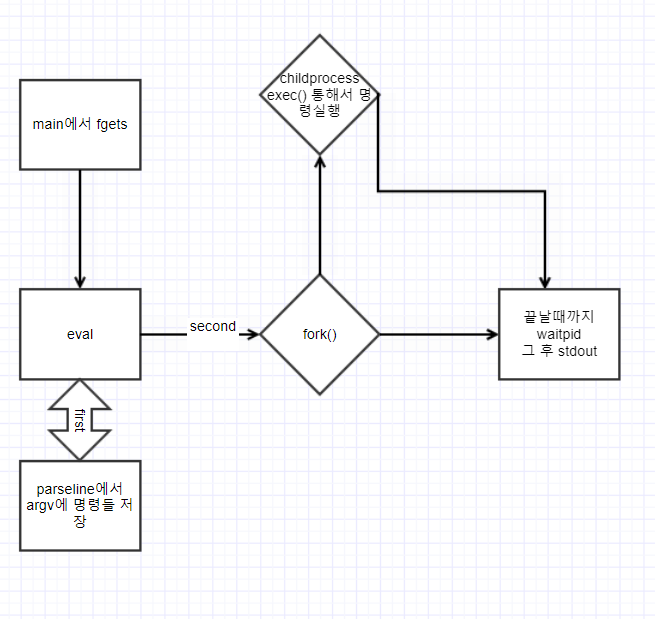
pipeline 개수마다 자식 프로세스가 돌려지기 때문에, 하나가 시행이 되면 그 하나가 멈출때까지 wait한뒤에 다음 자식 프로세스를 돌렸다.

* **Phase3 (background process)**
  + Background (’&’) process를 구현한 부분에 대해서 간략히 설명
  1. **개발 방법**

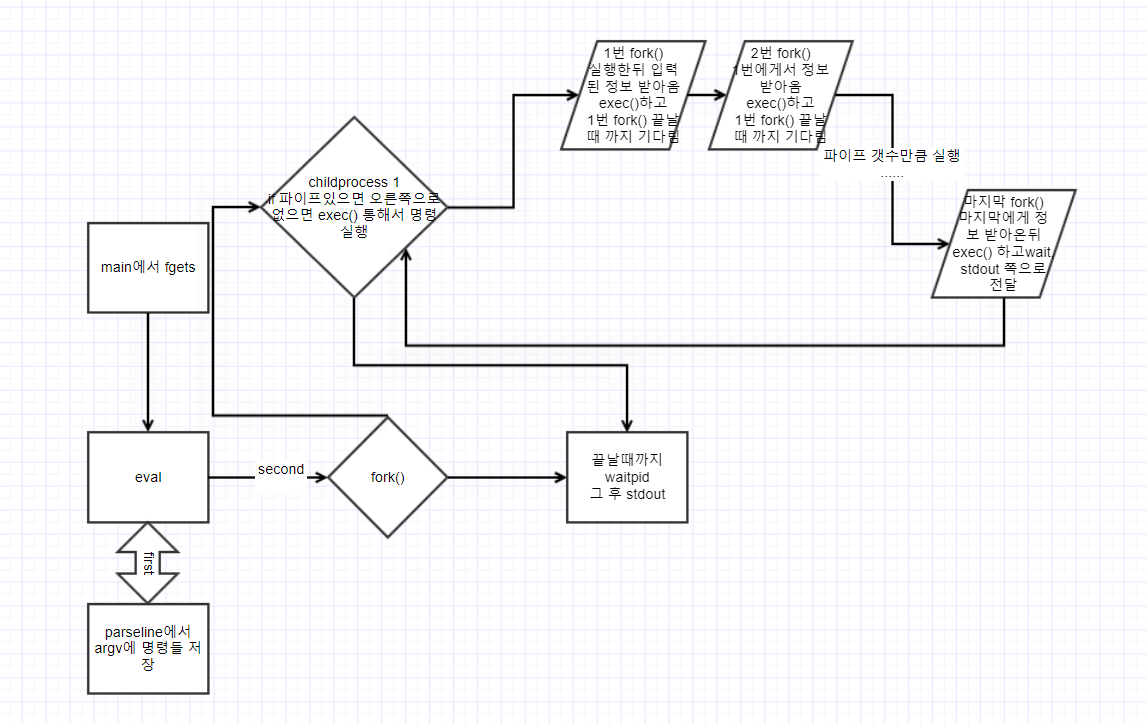
phase1 : 기본적으로 eval에서 fork()해야한다. fork()하기전 부모 프로세스 상태에서 cd를 구현하기 위해서, chdir함수를 구현하고, fork()를 한뒤에 자식프로세스에서 exec()를 통해서 주어진 셀 명령을 실행한다. fork()가 끝난 후에는 waitpid를 통해서 부모프로세서가 자식프로세서를 종료시킨다.  
추가로 ctrl+c를 구현하기 위해서 signal(SIGINT,함수)를 사용하였다.  
echo시 따옴표 처리를 위해서 parseline에 따옴표 아스키 코드를 이용해서 따옴표 있을 시 안에 blank가 들어와도 그대로 argv에 저장하도록 구현하였고(다음 따옴표 올때까지 buf를 밀어줌, 다음 따옴표전에 null 들어오면 되돌아감) 따옴표는 echo될때 제외되도록 구현하였다.

phase2 : phase1에서 fork되서 명령을 처리하는 부분의 함수를 새롭게 만들었다(run\_command) run\_command에서는 파이프가 얼마나 있는지 처리하고, 해당 파이브 위치를 기억하는 배열을 만든다. 파이프가 없다면 phase1 처럼 처리하고, 파이프가 있다면 go\_pipe()로 이동하게 되어서. 각각 처음 시작하는 파이프, 중간 파이프들, 마지막 파이프를 순서대로 구현하였다. 처음 시작하는 파이프는 stdout를 막아주어야 자연스럽게 흘러가고, 중간 파이프는 양쪽을 다 막아주어야 자연스럽게 흘러간다. 마지막 파이프는 stdin를 막음으로써 흘러가게 구현하였고, 각각의 자식 프로세서가 끝날때까지 기다려준 다음에 그 다음 자식 프로세서가 시작이 된다.

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

****

1. **Phase 2 (pipeline)**

****

1. **Phase 3 (background)**