**System Programming Project 1**

담당 교수 : 김영재 교수님

이름 : 이호성

학번 : 20171680

1. **개발 목표**

* **해당 프로젝트에서 구현할 내용을 간략히 서술.**
* **(MyShell을 만드는 전체적인 개요에 대해서 작성하면 됨.)**

이번 프로젝트에서는 linux 환경의 shell을 직접 구현하여 shell에 대한 이해도를 높이도록 한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

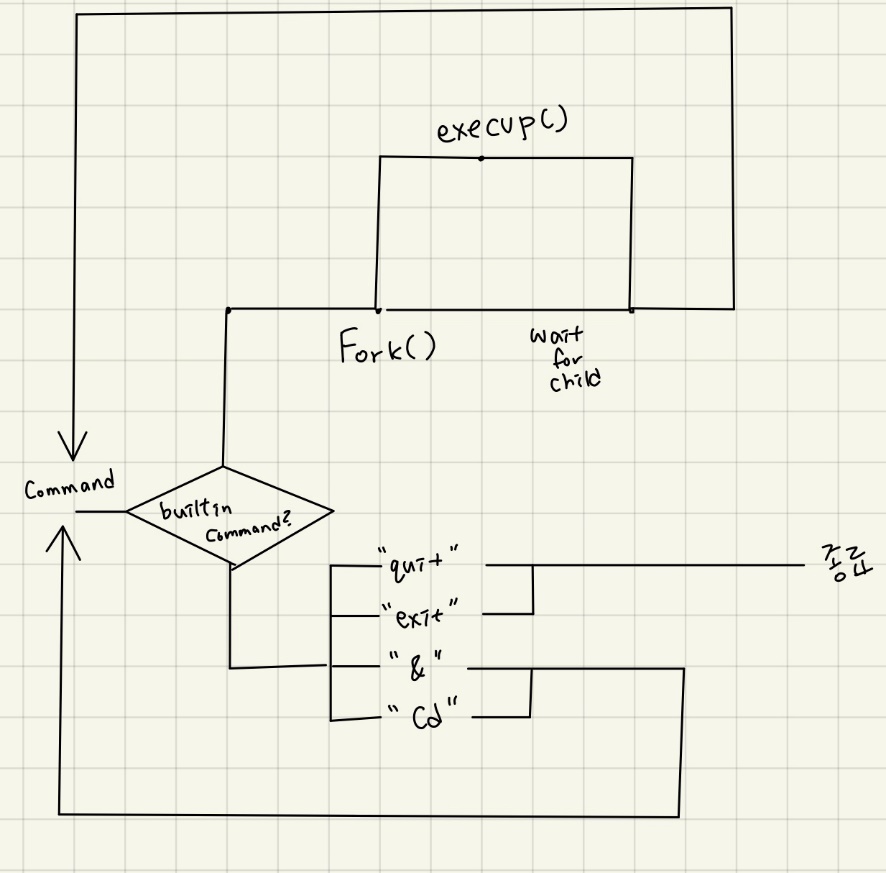
1. Phase 1 : 기본적인 shell의 기능을 수행하는 프로그램을 작성한다. exec 계열 함수를 사용하여 기본적인 명령어를 처리하고 cd, quit, exit 과 같은 built in command를 추가하여 해당 명령어들의 처리도 가능하게 한다.
2. Phase 2 : pipe 방식을 구현하여 명령어의 결과물이 STDOUT 이 아닌 다른 명령어의 input 값이 될 수 있도록 프로그래밍 한다.
3. Phase 3 : 백그라운드 작업의 개념을 추가하여 작업이 실행되는 도중에 다른 명령어를 받을 수 있도록 한다. 또한 백그라운드로 돌아가는 작업을 포그라운드 작업으로 올리거나 그 반대의 작업도 가능하게 하며 현재 돌아가고 있는 작업들을 확인 할 수 있는 job 명령어와 현재 돌아가고 있는 작업들을 중단 할 수 있는 kill 명령어 등을 처리할 수 있도록 한다.
   1. **개발 내용**

* **아래 항목의 내용만 서술**
* **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
* **Phase1 (fork & signal)**
  + fork() 를 통해 자식프로세스를 생성 하여 자식프로세스에서 execvp()를 통해 해당 명령어를 실행하도록 한다. 부모프로세스에서는 자식프로세스를 wait하며 자식 프로세스가 완료될 때 될 때 자식 프로세스가 부모 프로세스에게 Signal을 보내면 부모 프로세스가 이를 인지하고 wait을 종료한 후 다음 입력을 받게 된다.
* **Phase2 (pipelining)**
  + strtok() 함수를 사용하여 ‘|’ 문자를 기준으로 토큰을 나누어 한 명령어 씩 처리한다. 이때 한 명령어의 결과물이 STDOUT에 출력되는 것이 아닌 다른 명령어의 입력 값으로 활용 하기 위해 pipe()와 dup2() 함수를 사용하여 파이프와 버퍼를 통해 입력 값을 전달 한다. 이때 ‘|’ 의 개수를 미리 세어 반복문을 활용하여 명령어들을 처리한다.
* **Phase3 (background process)**
  + 명령어가 & 문자와 함께 입력이 된다면 백그라운드 작업으로 실행한다. 백그라운드 작업을 실행하는 방법은 Fork()를 실행하여 자식프로세스에서는 명령어를 실행하고 부모프로세스에서는 포그라운드 작업이라면 자식 프로세스를 wait하여 다음 명령어를 받지 못하게 기다리고 백그라운드 작업이라면 다음 명령어를 바로 받을 수 있도록 wait 하지 않고 진행한다. 이때 백그라운드 작업이라면 job list 에 추가하여 jobs 명령어를 입력 받으면 현재 실행 중인 프로세스들을 알 수 있도록 한다. 또한 kill 명령어를 통해 백그라운드 작업을 강제로 종료 할 수 도 있게 된다.
  1. **개발 방법**
* **B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**
* **Phase1**
  + 수업시간 주어진 shellex.c 의 코드를 기본 스켈레톤 코드로 활용하였다. 여기에 builtin\_command 에 exit을 추가해주었으며 cd 를 처리하기 위해 chdir() 함수를 추가하였다. 이외 명령어들을 처리하기 위해서 execvp() 함수를 사용하였으며 한번 입력을 받고 프로그램이 종료되지 않도록 Fork() 함수를 사용하여 자식 프로세스에서 execvp()를 활용하여 명령어를 처리하고 부모 프로세스에서 wait 한 다음 진행하여 명령어를 여러 번 받을 수 있도록 처리 하였다.
* **Phase2**
  + pipelining 을 처리할 수 있도록 ‘|’ 를 기준으로 strtok() 함수를 이용하여 명령어들을 토큰으로 나누어 처리하였다. Fork() 를 활용하여 한 명령어의 결과가 나오길 기다린 다음 결과가 저장된 buffer로 통하는 file descriptor를 저장한 다음, dup2() 함수를 활용하여 다음 명령어를 처리할 때 이전 명령어의 결과가 저장되어 있는 버퍼에서 입력 값을 받아들인다.
* **Phase3**
  + 기존 명령어를 처리하기 위한 Fork() 이전에 백그라운드와 포그라운드 작업을 처리하기 위한 Fork() 함수를 한번 더 사용한다. 포그라운드로 실행이 된다면 다음 입력을 받으면 안되므로 자식프로세스를 wait 해준다. 백그라운드로 실행 했다면 자식 프로세스를 기다리지 않고 다음 명령어를 받을 준비를 한다. 이때 현재 실행되고 있는 job list 를 알아야 하므로 부모프로세스에서 job list 에 명령어를 추가하고 자식프로세스가 종료 될 경우 job list 에서 삭제하도록 한다. 현재 실행중인 백그라운드 프로세스를 종료하기 위해 kill 명령어를 bulitin\_command에 추가하여 해당 프로세스 그룹 아이디를 가진 프로세스들에게 SIGKILL 시그널을 보내어 강제 종료 시킨다.
  + fg, bg 명령어와 ctrl + c ,ctrl + z signal은 구현하지 못했다.

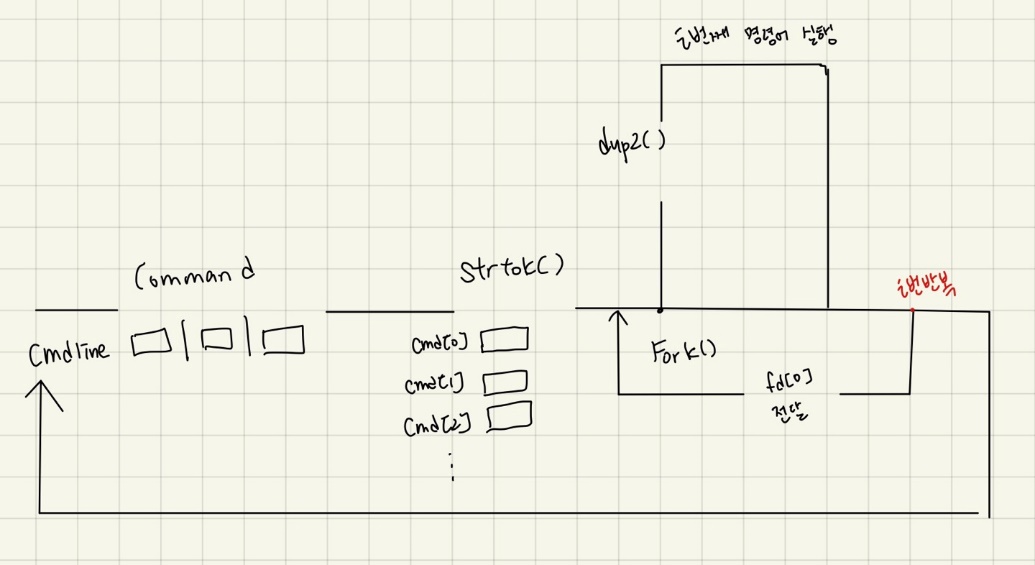
1. **구현 결과**
   1. **Flow Chart**

* **2.B.개발 내용에 대한 Flow Chart를 작성.**
* **(각각의 방법들에서 추가된 내용(fork, pipeline, background)만 특성이 잘 드러나게 그리면 됨.)**

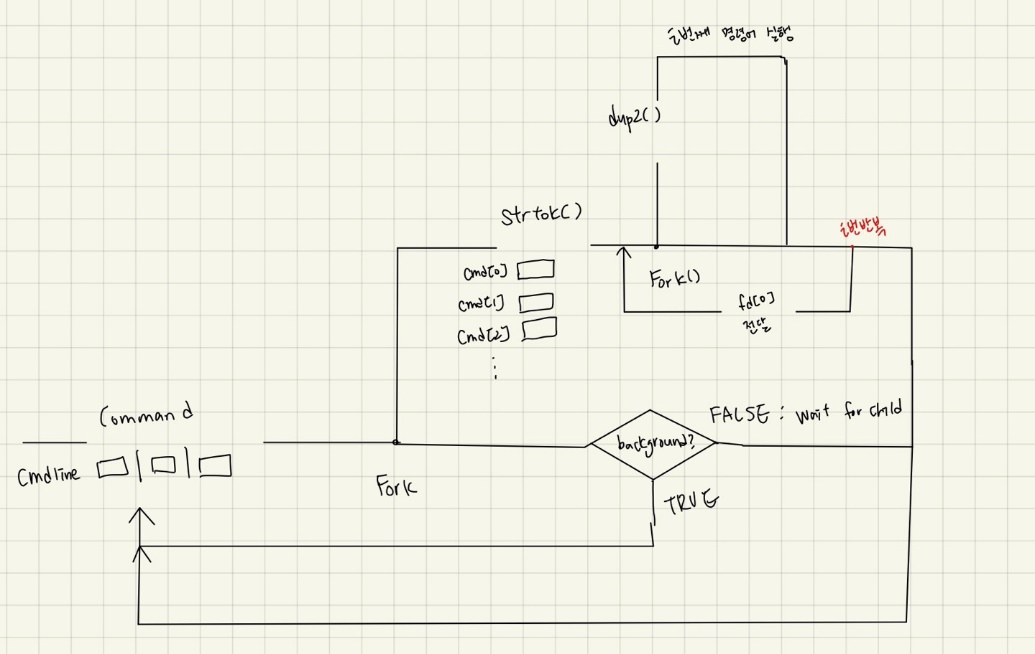
1. **Phase 1 (fork)**

****

1. **Phase 2 (pipeline)**

****

1. **Phase 3 (background)**

****