**Multicore Programming Project 1**

담당 교수 : 박성용 교수님

이름 :이용욱

학번 :20191626

1. **개발 목표**

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

c언어를 사용해서 myshell 을 구현한다. shell은 STDIN으로 명령을 입력받아 해당 명령을 수행한 후에 다시 반응 대기 상태로 돌아온다. myshell은 명령어 line을 parsing하여 built -in command를 수행할 지 또는 /bin/directory 상의 프로그램에 접근할 지 를 결정한다. /bin/directory상에서는 fork()를 통해 자식프로세스에서 그 명령어를 수행한다. 이번 프로젝트를 통해서 process signaling, system process control 그리고 history 명령어를 구현할 수 있다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Phase 1

cd, exit 등과 같은 기초 빌트인 커맨드와 /bin/directory에서 수행되는 shell프로그램을 구현한다. built in command외에는 foreground에서 동작한다.

1. Phase 2

파이프의 구현을 통해 여러 개의 명령어가 상호작용하도록한다. fd를 통해서 해당 명령어의 결과를 grep하여 다음 명령어의 입력으로 사용할 수 있도록한다.

1. Phase 3

job list와 터미널 관리를 통해서 bg processing을 할 수 있도록 구현한다.

* 1. **개발 내용**
* **아래 항목의 내용만 서술**
* **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
* **Phase1 (fork & signal)**
  + fork를 통해서 child process를 생성하는 부분에 대해서 설명
  + connection을 종료할 때 parent process에게 signal을 보내는 signal handling하는 방법 & flow

Fork():

입력된 command가 built in 이 아닌 경우에는 shell이 아닌 자식 프로세스가 execve()함수를 통해서 입력된 command를 수행한다. 이를 위해서 if 문에 pid값이 0인 자식프로세스가 해당 조건문에 들어가서 execve()함수를 호출할 수 있도록 개발하였다.

Signal: Myshell에서 자식프로세스가 fork되었을 경우 그 작업이 끝날 때까지 interactive모드로 돌아가면 안되는데, 끝난 자식 프로세스는 reaping해주어서 메모리 누수를 방지해야한다. 이는 wait()함수를 통해서 구현할 수 있다.

history : historylog.txt 파일을 통해 입력받은 명령어 리스트를 관리하고 이를 다시 사용할 수 있도록 하였다. 이때 !! 와 !#을 해당 위치의 명령어로 치환하는 방식으로 구현하여 명령어의 연결 또한 가능하도록 하였다.

* **Phase2 (pipelining)**
  + Pipeline( ‘|’ )을 구현한 부분에 대해서 간략히 설명 (design & implementation)
  + Pipeline 개수에 따라 어떻게 handling했는지에 대한 설명
* **Phase3 (background process)**
  + Background (’&’) process를 구현한 부분에 대해서 간략히 설명
  1. **개발 방법**
* **B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

헤더파일

<csapp.h> 교수님께서 제공해주신 파일 안에 여러가지 기본 헤더파일과 <sys/wait.h>와 같은 wait함수 등이 담겨져 있어 시스템 프로세스 구현에 활용하였다.

1)phase 1

eval()함수내에서 우선 history 관련 전처리 ( !! !#)를 해주어 historylog.txt파일에서 해당 명령어를 가져오고 실행 할 수 있도록 하였다. built in command가 아닌 경우에는 fork() 를 통해서 자식프로세스 내부에서 execve()함수를 통해 /bin/directory 에 저장된 프로그램을 실행하고 부모는 자식 프로세스의 종료시까지 대기한다.

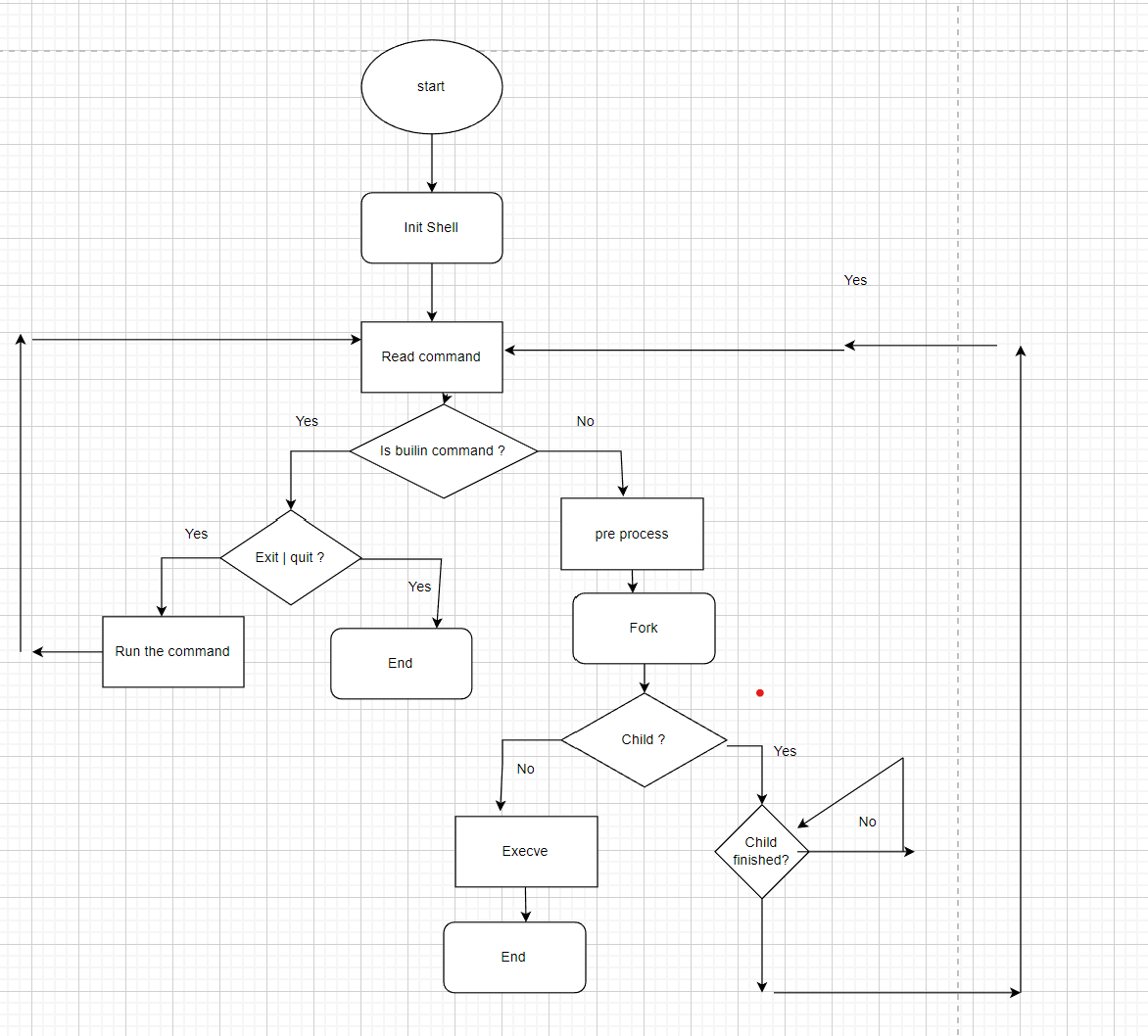
history 관련 함수 구현

history의 활용을 위해 우선 historylog.txt 를 a+모드로 open하여 입력되는 stdin명령어들을 읽고 쓸수 있도록 하였고, last\_command 라는 변수에 historylog.txt의 마지막 command를 담아 초기화 하고 이후 입력되는 명령어들과 비교하며 갱신해 주었다. 이와 같은 구현을 통해 history 기능을 완성할 수 있었다. !! -al 같은 경우처럼 히스토리 상의 명령어와 입력 명령어가 합쳐지는 경우에는 !!를 해당 명령어로 치환하는 방법을 통해 구현하였다.

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)**
2. **Phase 3 (background)**