**System Programming Project 1**

담당 교수 :김영재

이름 :강유석

학번 :20192021

1. **개발 목표**

Phase1

시스템 콜 함수인 exec() 계열의 함수를 조사해 process를 덮어쓰는 방법을 익힌다.

fork()를 통해 shell 명령어에 해당하는 process를 새로 띄워서

그 안에서 exec() 함수를 통해 명령을 수행하고 exit을 통해 다시 부모 process로

돌아와 마무리한다.

Phase2

Child process 를 2개 띄어 서로를 연결하는 pipe함수를 이용해 file descriptor를

조작해 출력과 입력을 연결합니다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Phase 1

시스템 내에 저장되 있는 함수들 (ls, pwd ps 등등) 제대로 나왔고

Built\_in 함수인 exit과 cd의 경우도 따로 함수를 작성해 해결하였다.

특히 cd의 경우 chdir함수를 수정해 구현하였다.

1. Phase 2

Pipe 함수를 이용해 첫번째 process의 명령어에 해당하는 내용이

두번째 process의 입력으로 들어가 이에 해당하는 출력 결과가

터미널 출력으로 바꿔서 나왔다.

1. Phase 3
   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

fork함수를 통해 return값을 pid에 저장해 pid = 0이면 child process

pid > 0이면 parent process , 나머지 경우에는 에러를 출력

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

먼저 parent process 부분에 wait함수를 사용해 child process가 exit 되기 전까지 기다렸다. Child process가 exit을 하면 wait하고 있던 parent process가

깨어나 child process를 reaping 한다.

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

Parsing 부분에서 두개의 buffer를 미리 만들어 입력된 command에서 “|” 가 발견되면 이전의 token을 pipe\_split1이라는 buffer에 저장하고

“|” 뒤에 나오는 명령은 pipe\_split2에 저장하였다.

* + Pipeline 개수에 따라 어떻게 handling했는지에 대한 설명

“|”를 만날 때 마다 이전의 명령들을 pipe\_split 이라는 buffer에 저장하였다.

* **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)**

Phase 1

명령어를 읽는 부분은 getchar() 함수를 사용하여 한 문자씩 읽어 들였습니다.

다 읽었으면 마지막에 null을 삽입해 해당 문자열을 return 해주었습니다.

Parsing() 부분에서는 strtok 함수를 사용해 “ ,.\n” 에 해당하는 문자를 만나면

torken으로 잘라주었습니다.

Torken으로 자른 명령어들은 argv에 넣어보관했습니다.

마지막으로 fork() 를 통해 child process에 도달하면 excvp를 통해 argv를 넣어

시스템 콜을 통해 수행하였습니다.

다 끝나면 child process는 exit을 통해 parent process에게 reaping 해달라고 신호를 보내고 parent process는 wait 된 상태가 풀리면서 return 하게 됩니다.

Phase 2

Phase1과 비슷하지만 pipe 처리를 따로 구현해주었습니다.

Phase1의 parsing() 함수에 “|”를 만나면 pipe\_split1에 “|” 만나기 전까지의 명령어를 넣어주었습니다. “|” 뒤에 나오는 명령어는 pipe\_split2에 넣어주었습니다.

Pipe\_action함수는 pipe를 처리하는 함수입니다.

Pipe() 함수를 이용해 pipe를 열어두고

dup2() 함수를 이용해 file descriptor의 input과 output을 pipe의 end point로

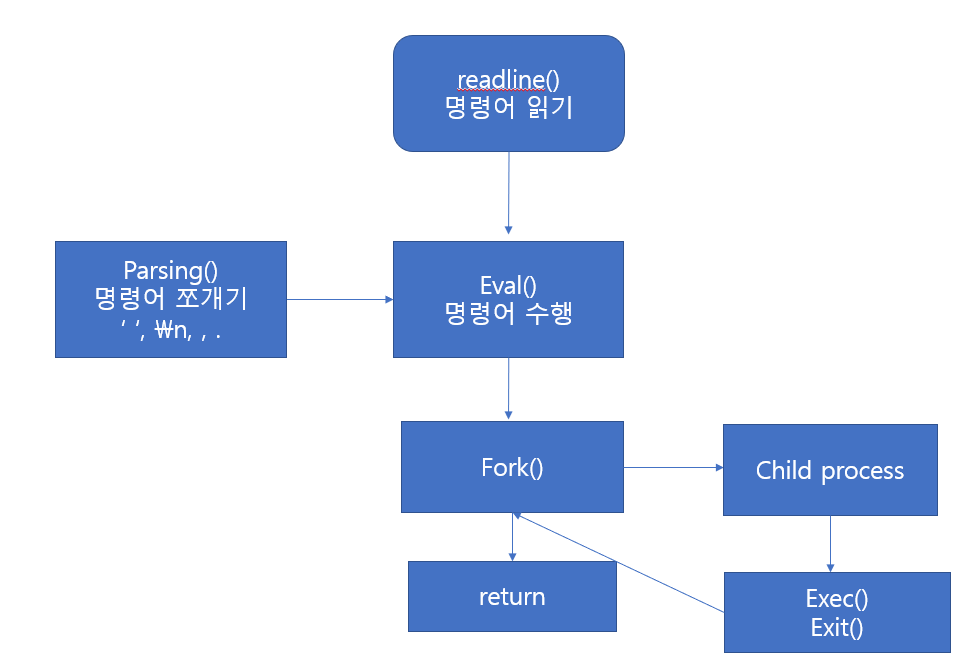
바꿔주었습니다.

명령어를 다 수행하면 close를 통해 pipe를 닫아줍니다.

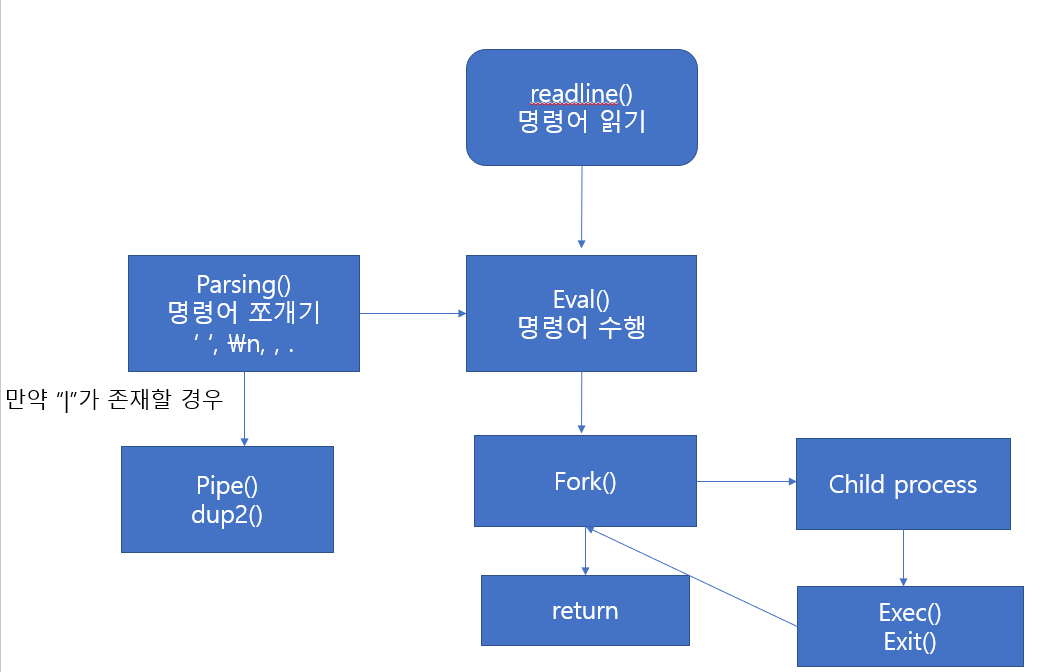
1. **구현 결과**
   1. **Flow Chart**

* **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)**