**Operating System HW2**

학번: 2016113832

이름: 윤동준

담당교수: 강순주

1. **개요**

Sum, Subtraction, Division, Factorial이라는 함수를 만들어서 argc와 argv[]를 통하여 command창에서 입력을 받아서 원하는 연산을 수행하는 실행파일(myshell.exe)을 만들었다.

1. **프로그램 구조 설명**

**2.1 함수에 대한 설명**

**sum:** int main(int argc, char\* argv[])로 커맨드창에서 값을 입력받으면 그 입력 받은 값을 argv[]에 저장을 한다. 그리고 저장한 값을 더해서 출력해준다.

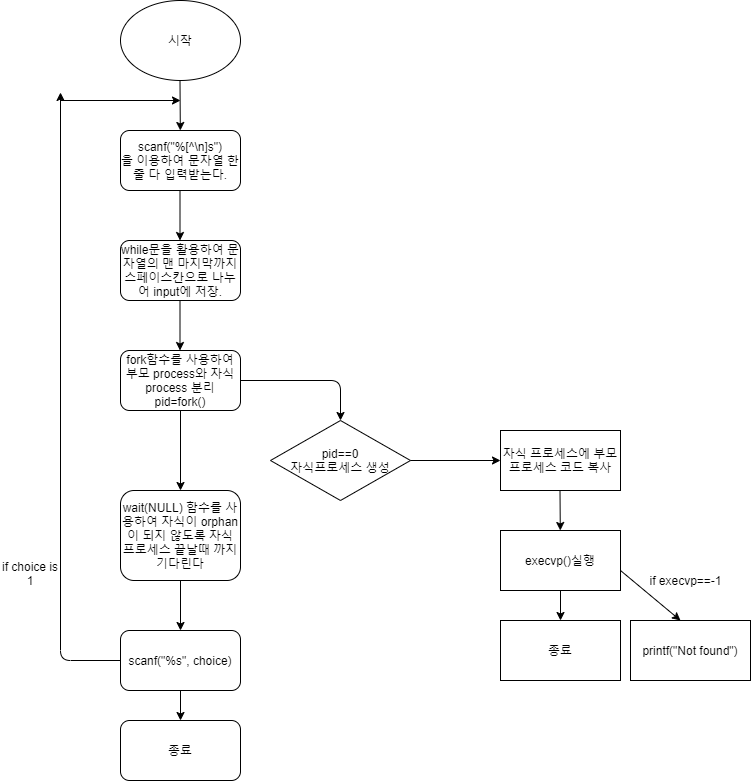
**sub:** int main(int argc, char\* argv[])로 커맨드창에서 값을 입력받으면 그 입력 받은 값을 argv[]에 저장을 한다. 만약 argv[1]>argv[2]보다 크면 그대로 argv[1]-argv[2]를 출력하고, argv[2]>argv[1]이면 argv[2]-argv[1]한 값을 출력해준다.

**div:** int main(int argc, char\* argv[])로 커맨드창에서 값을 입력받으면 그 입력 받은 값을 argv[]에 저장을 한다. 만약 argv[1]>argv[2]보다 크면 그대로 argv[1]/argv[2], argv[1]%argv[2]를 하고, argv[2]>argv[1]이면 argv[2]/argv[1], argv[2]%argv[1]한 값을 출력해준다.

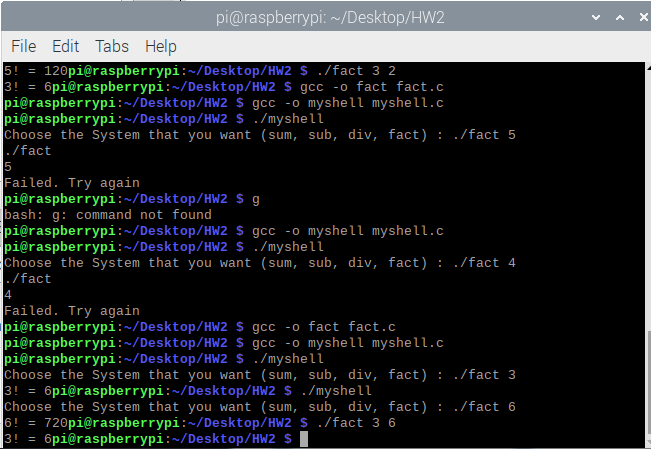
**factorial:** int main(int argc, char\* argv[])로 커맨드창에서 값을 입력받으면 그 입력 받은 값을 argv[]에 저장을 한다. 그리고 argv[1]에 저장된 값을 factorial해준다.

**myshell:** 처음 인자들을 scanf를 사용하여 한 줄 다 저장한 다음, strtok를 사용하여 공백을 기준으로 나누었다. if문을 사용하여 나눈것들을 처음에 선언해준 char\* input에다가 넣고 execvp를 활용하여 다른 실행파일을 실행하였다. 이때, execvp를 fork함수의 child process에 넣어서 실행하여, execvp로 인해 원래 process를 건들지 않게 한다. 그리고 wait(NULL)를 사용하여 child process가 끝날 때까지 대기한다. 그리고 마지막으로 while문을 사용하여 내가 원할 때 프로그램을 종료할 수 있게 하였다.

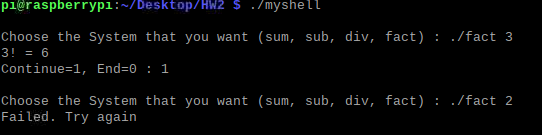
**2.2 다이어그램**



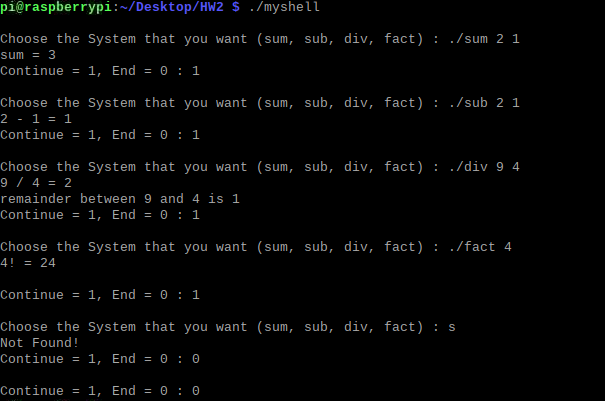
**3.실행 결과**

**오류**

메인함수에서 input[2]=NULL을 하였을 때 에러가 났다.



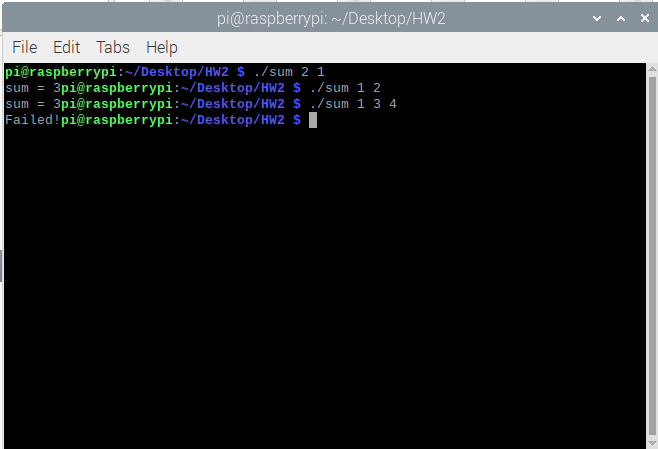
메인함수에서 while문을 사용하여 1번을 누르면 continue, 0번을 누르면 실행파일이 종료되는 것을 목표로 코드를 작성하였는데, 맨 마지막에 count를 초기화하지 않아서 오류가 났다.



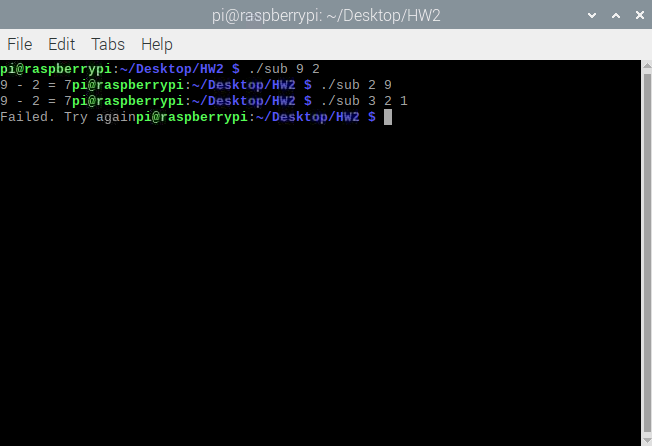
메인함수에 while문 맨 마지막 초기화하는 과정에서 getchar()를 count=0위에 적어서 이러한 오류가 났다.

**실행결과**

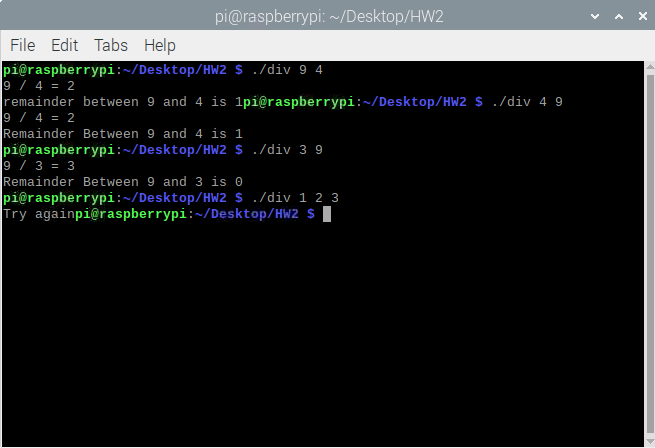
sum.exe



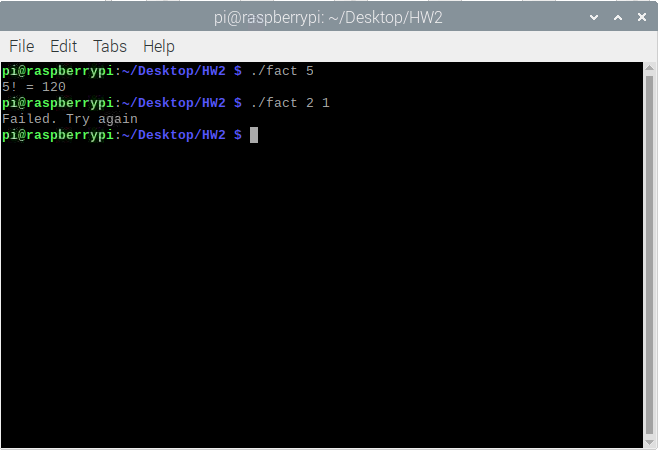
sub.exe



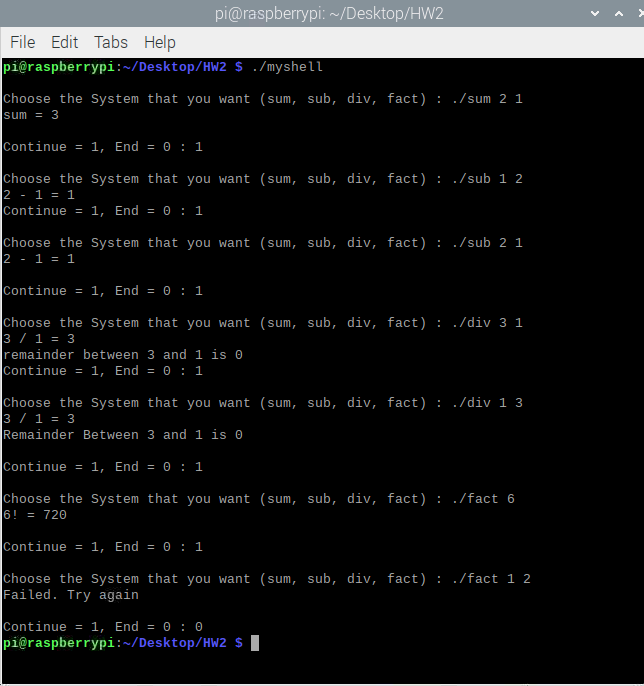
div.exe



fact.exe



­­myshell.exe



**4. 고찰**

처음 과제를 내주셨을 때, ‘에이 어려워봤자 얼마나 어렵겠어. 그냥 sum, subtraction, division, factorial c언어 코드 짜서 myshell 실행파일 만들면 되겠지’라고 생각했지만, int main(void) void에 int argc, char\* argv[]를 사용하라 하셔서 매우 당황스러웠다. c언어를 하면서 한 번도 사용하지 않아봐서 개념을 이해하기 힘들었지만, 구글에서 찾아보면서 command창에서 값을 입력받을 수 있다는 것을 배웠다. char\* argv[]를 통해 처음 HW1에서 공부했던 pointer에 대해 한 번 더 복습할 수 있어서 좋았다. 솔직히 말하면 아직 확실하게 포인터가 어떤 건지 감이 오진 않지만 그래도 포인터 쓰는 것이 친숙해지고 있다. sum, subtraction, division, factorial.c 파일을 compile해서 나온 cmd창이 아닌 컴퓨터 Terminal창에서 폴더를 열고 실행해서 값을 넣었을 때 연산이 된다는 것이 너무 신기했다. 첫 번째 문제를 해결하고 다음 문제로 넘어가려고 하니, myshell.exe를 만들어서 1번을 하라고 하는데 도대체 어떻게 해야 할지 갈피를 못 잡아서 인터넷을 찾아보고 감을 잡았다. 메인 프로그램에서 인자를 입력받아서 어떻게 나눌까 고민을 해봤는데, strtok를 이용하여 공백 한 칸 기준으로 나누어서 배열에 넣어줬다. 처음 쉘 프로그램을 짜는데 fork함수가 왜 필요한지 몰랐는데, execvp함수 때문에 필요하다는 걸 알게 되었다. execvp함수가 실행이 되면, 지금 프로세스에다가 덮어쓰는 것이니, 밑에는 실행이 되지 않는다. 그래서 fork를 사용하여 child 프로세스에다가 execvp함수를 넣어서 실행시켰다. 프로그램이 한번 실행하고 끝내지 않기 위해 while문을 사용하여 내가 원하는 때에 프로그램을 끌 수 있게 하였다. execvp(const char\* file, char\* const argv[])는 file을 실행하고 argv[]에 인자값을 넣는다. makefile같은 경우에는 HW1에서 만들어 놓은 매크로를 변경하여 사용하였다. 운영체제를 처음 시작할 때는 #include <stdio.h>도 제대로 못 쳤었는데 과제를 하나하나 수행하면서 뿌듯함을 많이 느끼고 있다.

**5.** **프로그램 소스 파일**

**sum.c**

#include<stdio.h>

#include<stdlib.h> // to use atoi function

int main(int argc, char\* argv[]){

int i;

int sum;

if(argc!=3){

printf("Failed!");

return 0;

}

sum= atoi(argv[1])+atoi(argv[2]);

printf("sum = %d", sum);

return 0;

}

**sub.c**

#include <stdio.h>

#include <stdlib.h> // to use atoi function

int main(int argc, char\* argv[]) {

int sub=0;

if (argc != 3)

{

printf("Failed. Try again"); //if you type more or less then three string, it will apear "Try again".

return 0;

}

if (atoi(argv[1])>atoi(argv[2])) //if the number which argv[1] is greater then argv[2], then sub argv[1]-argv[2]

{

sub= atoi(argv[1])-atoi(argv[2]);

printf("%s - %s = %d\n",argv[1], argv[2], sub);

}

else { //if the number which argv[2] is greater then argv[1], then sub argv[2]-argv[1]

sub= atoi(argv[2])-atoi(argv[1]);

printf("%s - %s = %d\n", argv[2], argv[1], sub);

}

return 0;

}

**div.c**

#include <stdio.h>

#include <stdlib.h> // to use atoi function

int main(int argc, char\* argv[]) {

int div=0;

int remainder=0;

if (argc != 3){

printf("Try again"); //if you type more or less then three string, it will apear "Try again".

return 0;

}

if (atoi(argv[1]) > atoi(argv[2])) //if the number which argv[1] is greater then argv[2], then div argv[1]/argv[2] and argv[1]%argv[2]

{

div= atoi(argv[1])/atoi(argv[2]);

remainder= atoi(argv[1]) % atoi(argv[2]);

printf("%s / %s = %d\n", argv[1], argv[2], div);

printf("remainder between %s and %s is %d", argv[1], argv[2], remainder);

}

if (atoi(argv[2])> atoi(argv[1])) //if the number which argv[2] is greater then argv[1], then div argv[2]/argv[1] and argv[2]%argv[1]

{

div= atoi(argv[2])/atoi(argv[1]);

remainder= atoi(argv[2]) % atoi(argv[1]);

printf("%s / %s = %d\n", argv[2], argv[1], div);

printf("Remainder Between %s and %s is %d\n", argv[2], argv[1], remainder);

}

return 0;

}

**fact.c**

#include <stdio.h>

#include <stdlib.h> //to use atoi function

int main(int argc, char\* argv[]) {

int i=0;

int fact=1;

if (argc != 2){

printf("Failed. Try again\n"); //if you type more or less then two string, it will apear "Try again".

return 0;

}

for (i=0; i<atoi(argv[1]); i++) { //by using atoi function, I made string to integer, and then run for until atoi

fact=fact\*(atoi(argv[1])-i);

}

printf("%s! = %d\n", argv[1], fact);

return 0;

}

**myshell.c**

#include <stdio.h>

#include <unistd.h> //To use fork() and exec()

#include <sys/wait.h> //To use wait()

#include <string.h> //To use strcmp()

int main (void) {

char choice[10];

char\* input[10]; //because of the execvp format, you need to get the same input compare to int main(int argc, char\* argv[]).

int count=0;

char\* practice;

int pid;

while(strcmp(choice, "0")!=0){

printf("\nChoose the System that you want (sum, sub, div, fact) : ");

scanf("%[^\n]s", choice); //you get the all the character except enter

getchar(); //to clear the buffer

practice=strtok(choice," ");

while(practice != NULL){

input[count] = practice;

practice= strtok(NULL, " "); //dividing by space

count++;

}

input[count]=NULL; //if you use execvp, at the end you should put null

pid = fork();

if(pid == 0){

if(execvp(input[0], input) == -1) {

printf("Not Found!"); //input is same as char\* argv[].

}

}

wait(NULL); //until child process end wait

printf("\nContinue = 1, End = 0 : ");

scanf("%s", choice);

count=0;

getchar();

}

return 0;

}

**6. 자료 출처**

1. [https://dojang.io/mod/page/view.php?id=696](http://dojang.io/mod/page/view.php\?id=696) (argc,argv)

2. [https://jhnyang.tistory.com/57](http://jhnyang.tistory.com/57) (Shell)

3. [https://dojang.io/mod/page/view.php?id=376](http://dojang.io/mod/page/view.php\?id=376) (strtok)

4. [https://12bme.tistory.com/223](http://12bme.tistory.com/223) (exec, fork, wait)

5. [https://channelofchaos.tistory.com/55](http://channelofchaos.tistory.com/55) (exec, fork)