**강의명: 유닉스 시스템**

**실습 번호: 6**

**실습 제목: 프로세스 환경(Process Environment)**

**학생 이름: 황귀훈**

**학번: 201710885**

**1. 프로세스 종료 값**

**1.1**

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

**int exitno = atoi(argv[1]);**

**fprintf(stdout,"exit=%d\n",exitno);**

**exit(exitno);**

**}**

1번 프로세스 종료 값은 argv[1]에 주어지는 정수를 종료 값으로 종료하는 프로그램이다. 이 프로그램은 종료하기 전에 형식 “exit=%d\n”을 사용하여 종료 번호를 표준 출력에 출력하여야 한다.

프로그램 처음에 헤더파일을 include해줘야 하는데 atoi()함수와 exit()함수를 사용하기 위해 <stdlib.h>를, fprintf()함수를 사용하기 위해 <stdio.h>를 include하였다.

프로그램의 목적에 맞게 먼저 argv[1]에 주어지는 string값을 atoi()함수를 사용하여 정수 값으로 바꾼뒤 exitno변수에 저장했다. 그리고 fprintf()함수를 사용하여 표준 출력에 주어진 형식에 맞게 출력한 뒤 프로그램을 종료하기 위해 exit()함수에 exitno를 인수로 주어 프로그램을 종료했다.

**1.2**

****

**2. 프로세스 종료 처리 함수**

**2.1**

**#include <stdio.h>**

**#include <stdlib.h>**

**int exitno;**

**void printExitNo(void)**

**{**

**fprintf(stdout,"exit=%d\n",exitno);**

**}**

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

**{**

**atexit(printExitNo);**

**exitno = atoi(argv[1]);**

**exit(exitno);**

**}**

2번 프로세스 종료 처리 함수는 argv[1]에 주어지는 숫자를 종료 값으로 종료하는 프로그램으로 이 프로그램은 종료하기 전에 atexit 처리 함수가 형식 “exit=%d\n”을 사용하여 종료 번호를 출력해야 한다.

먼저 헤더파일을 include 해야 하는데 atexit()함수와 atoi()함수 그리고 exit()함수를 사용하기 위해 <stdlib.h>를, fprintf()함수를 사용하기 위해 <stdio.h>를 include하였다.

그리고 종료 번호를 저장할 변수 exitno를 선언한 뒤 형식 “exit=%d\n”을 사용하여 종료 번호를 출력할 함수 **printExitNo(void)**을 작성하였다, **printExitNo(void)**함수에서는 간단하게 fprtinf()함수를 사용하여 형식에 맞게 종료 번호 exitno을 출력한다.

main()함수 에서는 먼저 atexit()함수를 호출하는데 exit()함수가 호출되었을 때 수행할 함수 로 위에서 작성한 **printExitNo(void)**함수를 등록했다. 그리고 argv[1]에 주어지는 string값을 atoi()함수를 사용하여 정수 값으로 바꾼뒤 exitno변수에 저장했다. 마지막으로, 프로그램을 종료하기 위해 exit()함수에 exitno변수를 인수로 주어 프로그램을 종료했다. 따라서 exit(exitno)함수가 호출돼 프로그렘이 종료하기 전에 atexit() 처리 함수가 형식 “exit=%d\n”을 사용하여 종료 번호를 출력하고 argv[1]에 주어지는 숫자를 종료 값으로 프로그램이 정상 종료된다.

**2.2**

****

**3. 명령 줄 인수**

**3.1**

**#include <stdio.h>**

**#include <stdlib.h>**

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

**{**

**int sum = 0;**

**for(int i = 1; i < argc; i++) {**

**fprintf(stdout,"%d ",atoi(argv[i]));**

**sum += atoi(argv[i]);**

**}**

**fprintf(stdout,"\nsum=%d\n",sum);**

**exit(0);**

**}**

3번 명령 줄 인수는 인수로 주어지는 1개 이상의 정수들을 화면의 첫 줄에 순서대로 출력하고 그 값들의 합을 다음 줄에 출력하는 프로그램이다.

먼저 exit()함수와 atoi()함수를 사용하기 위해 <stdlib.h>를, fprintf()함수를 사용하기 위해 <stdio.h>를 include하였다.

먼저 인수로 주어지는 값들의 합을 저장할 변수 sum을 선언한 뒤 for문을 사용하여 반복적으로 인수로 주어지는 값들을 출력하고 sum에 더해지도록 프로그램을 작성하였다. 이때 for문을 사용하여 각 인수들을 계산할 때 atoi()함수를 사용해 string형식으로 들어오는 인수들을 int형으로 변환하고, 인수들을 화면의 첫 줄에 순서대로 출력하기 위해 fprintf()함수를 사용해 출력하였고 그 때마다 sum변수에 인수로 주어지는 정수들을 더하여 합을 계산했다. 주어지는 인수의 개수는 argc에 저장되기 때문에 이것을 사용하여 for문의 반복횟수를 결정했다.

화면에 인수를 출력하고 인수들의 합을 구하는 과정이 끝나면, 인수들의 합이 저장된 sum변수 값을 출력하기 위해 fprintf()함수를 사용하여 형식 "\nsum=%d\n"에 맞게 출력했다. 그 후 exit(0)함수를 사용하여 프로그램을 정상 종료했다.

**3.2**

****

**4. 환경 리스트**

**4.1**

**#include <stdio.h>**

**#include <stdlib.h>**

**extern char \*\*environ;**

**int main(void)**

**{**

**for (char \*\*env = environ; \*env; env++) {**

**fprintf(stdout, "%s\n", \*env);**

**}**

**exit(0);**

**}**

4번 환경 리스트는 현재 프로세스 수행 시의 환경 리스트를 출력하는 프로그램이다.

먼저. exit()함수를 사용하기 위해 <stdlib.h>를, fprintf()함수를 사용하기 위해 <stdio.h>를 include하였다.

그리고 현재 프로세스의 environment list에 접근하여 현재 프로세스 수행 시의 환경 리스트를 출력하기 위해서 char \*\* 형을 가지는 전역 변수 environ를 선언했다. 변수 environ는 환경변수 리스트의 포인터이다.

그 후 for문을 사용해 다음 포인터로 넘어 가면서 이 변수가 참조하는 string이 있으면 fprintf()함수를 통해 표준 출력으로 그 값을 출력할 수 있게 하였다. 환경리스트를 모두 출력하면 exit(0)함수를 사용하여 프로그램을 정상 종료하였다.

**4.2**

**s1710885@oak:lab06$ ./env**

**LC\_PAPER=ko\_KR.UTF-8**

**LC\_ADDRESS=ko\_KR.UTF-8**

**XDG\_SESSION\_ID=40877**

**LC\_MONETARY=ko\_KR.UTF-8**

**TERM=xterm**

**SHELL=/bin/bash**

**HISTSIZE=10000**

**…**

**XDG\_DATA\_DIRS=/usr/local/share:/usr/share:/var/lib/snapd/desktop**

**SSH\_CONNECTION=117.16.123.100 50953 117.16.47.164 22022**

**XDG\_RUNTIME\_DIR=/run/user/3019**

**LC\_TIME=ko\_KR.UTF-8**

**HISTTIMEFORMAT=%Y-%m-%d %T**

**HISTFILE=/etc/hpot/s1710885**

**LC\_NAME=ko\_KR.UTF-8**

**\_=./env**

**OLDPWD=/home/s1710885**

**5. 비 지역 점프**

**5.1**

**#include <stdio.h>**

**#include <setjmp.h>**

**jmp\_buf buf;**

**void second(void)**

**{**

**printf("second: entering...\n");**

**longjmp(buf, 1);**

**printf("second: exiting...\n");**

**}**

**void first(void)**

**{**

**printf("first: entering...\n");**

**second();**

**printf("first: exiting...\n");**

**}**

**int main()**

**{**

**printf("main: entering...\n");**

**if (setjmp(buf) == 0)**

**first();**

**printf("main: exiting...\n");**

**return 0;**

**}**

먼저 함수 헤더 부분 설명이다. setjmp()함수와 longjmp()함수를 사용하기 위해 <setjmp.h>를, printf()함수를 사용하기 위해 <stdio.h>를 include하였다.

함수 setjmp는 현재 지점을 jmp\_buf형 변수 buf에 저장하고 직접 불렸을 때는 0, longjmp에 의하여 돌아왔을 때는 non-zero값을 반환한다. 함수 longjmp는 이전에 저장된 jmp\_buf형 변수로 제어를 이동한다.

파일 setjmp.c에 작성된 프로그램은 setjmp와 longjmp함수를 사용해 제어(control)의 이동을 보여주는 프로그램이다. main()함수 에서는 먼저 printf()함수를 사용하여 “main: entering…\n”을 출력한다. 그리고 if문을 사용하는데 setjmp()함수를 사용해 jmp\_buf형 변수 buf에 현재 지점을 저장하고 처음 setjmp가 직접 호출되었는지를 판단하여 직접 불렸을 경우에는 first()함수를 호출한다. 그 후 printf()함수를 사용하여 “main: exiting…\n”을 출력하고 return 0로 프로그램을 종료한다. first()함수에서는 먼저 “first: entering…\n”을 출력하고 second()를 호출하고, “first: exiting…\n”을 출력한다. second()함수에서는 먼저 “second: entering…\n”을 출력하고 longjmp()함수를 호출한 뒤 “second: exiting…\n”을 출력한다. 이 때 longjmp()함수는 다시 제어를 main()에서 setjmp()가 호출되었던 부분으로 옮긴다. 따라서 처음 main()함수를 실행시키면 “main: entering…\n”가 출력되고 처음 setjmp()를 호출하면 반환 값이 0이기 때문에 first()함수를 호출하고 “first: entering…\n”가 출력된다. 그후 second()를 호출하면 “second: entering…\n”가 출력되고 longjmp()에 의해 처음 main함수에서 setjmp()를 호출했던 곳으로 제어가 옮겨져서 “first: exiting…\n”과 “second: exiting…\n”는 출력되지 않고 longjmp()에 의해 돌아왔을 때는 setjmp()가 non-zero값을 반환하기 때문에 “main: exiting…\n”가 출력되는 것이다.

**5.2**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

텍스트, 화이트보드이(가) 표시된 사진

자동 생성된 설명텍스트, 화이트보드이(가) 표시된 사진

자동 생성된 설명

program에는 local jump와 non-local jump가 있는데 둘의 차이는 control의 이동이 같은 함수내에서 이루어지는지 다른 함수로 control의 이동이 일어나는 지의 차이이다. Local jump는 같은 함수 내에서 control이동이 일어나기 때문에 call & return으로 main으로 가면 왼쪽 사진과 같이 새로운 local변수가 새로운 stack에 저장되어 이전 local변수 값과는 달라진다. 반면에 오른쪽 사진과 같은 non-local jump는 다른 함수로 가면서 그 사이에 쌓여 있던 stack을 다 버려야(pop)한다.

번호5번 비 지역 점프의 stack은 오른쪽 사진과 같은 모습이다. 따라서 “./setjmp”의 결과가 위와 같이 나오는 이유는 제일 처음 main()함수가 호출**(stack에 main() push)**되어 “main: entering…\n”를 출력하고 setjmp()함수를 호출해서 jmp\_buf형 변수 buf에 현재 지점**(빨간 화살표)**을 저장하고, first()함수를 호출한다. **(stack에 first() push)** first()함수에서는 "first: entering...\n”을 출력한 후 second()함수를 호출한다. **(stack에 second() push)** 그리고 second()가 호출되면 “second: entering…\n”가 출력되고 longjmp()에 의해 처음 main함수에서 setjmp()를 호출했던 곳**(stack에 빨간 화살표)**으로 제어가 옮겨지는데 이때 stack에 쌓인 first()와 second()가 pop되어서 “first: exiting…\n”과 “second: exiting…\n”는 출력되지 않고 longjmp()에 의해 돌아왔을 때는 setjmp()가 non-zero값을 반환하기 때문에 “main: exiting…\n”가 출력되는 것이다.