

과제2

-시스템콜을 통해 특정 기능을 수행-

syscall_64.tbl	시스템콜 테이블	/usr/src/linux/linux-5.15.120/arch/x86/entry/syscalls
syscalls.h	시스템콜 헤더파일	/usr/src/linux/linux-5.15.120/include/linux
Makefile	시스템콜 Makefile	/usr/src/linux/linux-5.15.120/kernel
sys_print_reverse.c	시스템콜함수: 자릿수 역순	/usr/src/linux/linux-5.15.120/kernel
sys_print_reverse2.c	시스템콜함수: 덧셈 뺄셈	/usr/src/linux/linux-5.15.120/kernel
assignment2.c	과제2를 구현하는 코드	/home/id20212211/

학번: 20212211

이름: 권대호

목차

1. 자릿수 역순 변경 시스템콜 구현 코드	3
2. 부호 바꾸어 덧셈, 뺄셈 연산 코드	5
3. Makefile, headerFile, assignment2.c 코드	6

1. 자릿수 역순 변경 시스템콜 구현

가. 소스코드: sys_print_reverse.c

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

//strlen과 같은 기능을 하도록 구현
int custom_strlen(const char *str) {
    int length = 0;
    while (str[length] != '\0') {
        length++;
    }
    return length;
}

//문자열을 뒤집는 함수 구현
void custom_reverse(char *str) {
    int length = custom_strlen(str);
    int start = 0;
    int end = length - 1;

    while (start < end) {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        start++;
        end--;
    }
}

//atoi 함수 구현
long custom_atoi(const char *str) {
    long result = 0;
    int sign = 1;

    while (*str == ' ') {
        str++;
    }

    if (*str == '-') {
        sign = -1;
        str++;
    } else if (*str == '+') {
        str++;
    }

    while (*str >= '0' && *str <= '9') {
        result = result * 10 + (*str - '0');
        str++;
    }
}
```

```

        return sign * result;
    }

SYSCALL_DEFINE1(print_reverse, long, inputValue) {
    char inputValueString[10000];
    sprintf(inputValueString, "%ld", inputValue);

    custom_reverse(inputValueString);

    long outputValue = custom_atoi(inputValueString);
    return outputValue;
}

```

나. 실행결과

```

id20212211@ubuntu:~$ ./assignment2.out
Input: 123
Output: 321
Input: 2345
Output: 5432
Input: 34567
Output: 76543
Input: 456789
Output: 987654
Input: 

```

다. 추가설명

/kernel에선 <string.h>, <stdio.h> 등 헤더파일을 읽을 수 없어서 특정 기능을 수행하는 함수(atoi, strlen)를 추가로 구현함.

custom_atoi는 우선 숫자가 str[]배열에 들어있을 때 부호가 붙어있는지 먼저 검사하고, 0과 9사이에 있는 수라면 반복할때마다 10씩 제공해서 result에 더하는 방식으로 작동한다.

custom_strlen은 \0을 만나기 전까지 length값을 1씩 증가시킨다.

문자열을 뒤집는 시스템콜은 sys_print_reverse(long inputValue)이고 뒤집은 long 값을 반환한다.

주어진 long inputValue의 길이를 length로 저장한 후 맨 앞과 뒤를 교환한다. 이때 배열 앞부분은 start, 뒷부분은 end로 하고 루프를 돌때마다 start++, end--과정을 거치고, 루프는 start<end까지 반복하기 때문에, 교환된 char가 한번 더 교환되지 않는다.

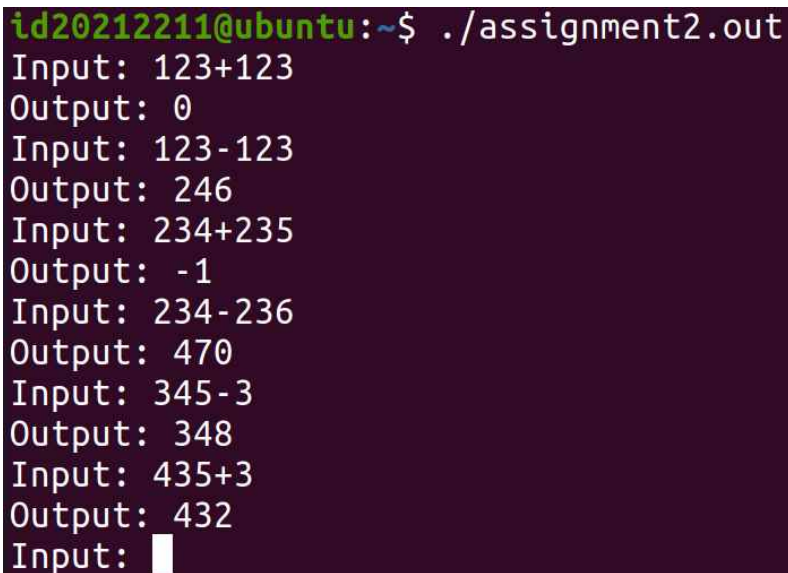
2. 부호 바꾸어 덧셈, 뺄셈 연산 코드

가. 소스코드: sys_print_reverse2.c

```
#include <linux/kernel.h>
#include <linux/syscalls.h>

SYSCALL_DEFINE3(print_reverse2, long, inputValueFirstLong, long, inputValueLastLong, int,
                operators) {
    long outputValueLong;
    if (operators == 1) {
        outputValueLong = inputValueFirstLong + inputValueLastLong;
    } else {
        outputValueLong = inputValueFirstLong - inputValueLastLong;
    }
    return outputValueLong;
}
```

나. 실행결과



```
id20212211@ubuntu:~$ ./assignment2.out
Input: 123+123
Output: 0
Input: 123-123
Output: 246
Input: 234+235
Output: -1
Input: 234-236
Output: 470
Input: 345-3
Output: 348
Input: 435+3
Output: 432
Input: 
```

다. 추가설명

sys_print_reverse2(long inputValue, long outputValue, operators)로 작동하는 시스템콜은 operators값이 1이면 덧셈, 0이면 뺄셈을 한다. 계산과정은 단순히 입력받은 inputValue와 outputValue를 operators값에 따라 더하거나 뺀 다음 반환한다.

3. 시스템테이블 MakeFile, HeaderFile, assignment2.c 코드

가. 소스코드: syscall_64.tbl

```
#
# 64-bit system call numbers and entry vectors
#
# The format is:
# <number> <abi> <name> <entry point>
#
# The __x64_sys_*() stubs are created on-the-fly for sys_*() system calls
#
# The abi is "common", "64" or "x32" for this file.
#
0      common  read              sys_read
1      common  write             sys_write
...
449    common  print_hello       sys_print_hello
450    common  print_reverse     sys_print_reverse
451    common  print_reverse2    sys_print_reverse2
#
# Due to a historical design error, certain syscalls are numbered differently
...
```

나. 소스코드: syscalls.h

```
/* SPDX-License-Identifier: GPL-2.0-only */
/*
 * syscalls.h - Linux syscall interfaces (non-arch-specific)
 *
 * ...
asmlinkage long sys_print_hello(void);
asmlinkage long sys_print_reverse(long inputValue);
asmlinkage long sys_print_reverse2(long inputValueFirstLong, long inputValueLastLong, int
operators);

#endif
```

다. 소스코드: Makefile

```
# SPDX-License-Identifier: GPL-2.0
...
obj-y    = fork.o exec_domain.o panic.o \
          cpu.o exit.o softirq.o resource.o \
          sysctl.o capability.o ptrace.o user.o \
          signal.o sys.o umh.o workqueue.o pid.o task_work.o \
          extable.o params.o \
```

```
kthread.o sys_ni.o nsproxy.o \  
notifier.o ksysfs.o cred.o reboot.o \  
async.o range.o smpboot.o ucount.o regset.o sys_print_hello.o  
sys_print_reverse.o sys_print_reverse2.o  
...
```

라. 소스코드: assignment2.c

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <linux/kernel.h>  
#include <sys/syscall.h>  
#include <ctype.h>  
  
int main(){  
    int shutDown = 1;  
  
    while (shutDown){  
        int program = 1;  
        char inputValue[10000];  
        int operators;  
  
        printf("Input: ");  
  
        if (fgets(inputValue, sizeof(inputValue)-1, stdin) == NULL){  
            break;  
        }  
  
        if (inputValue[0] == '\n'){  
            shutDown = 0;  
            break;  
        }  
  
        inputValue[strlen(inputValue) - 1] = '\0';  
        int length = strlen(inputValue);  
  
        for (int i = 0; i < length; i++){  
            if (inputValue[i] == '+'){  
                program = 2;  
                operators = 0;  
            }  
            else if (inputValue[i] == '-'){  
                program = 2;  
            }  
        }  
    }  
}
```

```

        operators = 1;
    }
    else if (!isdigit(inputValue[i])){
        program = 0;
        break;
    }
}

if (program == 1){
    long inputValueLong = atoi(inputValue);
    long outputValue = syscall(450, inputValueLong);
    printf("Output: %ld\n", outputValue);
}
else if (program == 2){
    char inputFirst[1000];
    char inputLast[1000];
    int i;
    int firstLength = 0;

    for (i = 0; i < length; i++) {
        if (isdigit(inputValue[i])) {
            inputFirst[i] = inputValue[i];
            firstLength++;
        }
        else {
            break;
        }
    }
    inputFirst[firstLength] = '\0';

    for (int j = i + 1; j < length; j++) {
        if (isdigit(inputValue[j])) {
            inputLast[j - i - 1] = inputValue[j];
        }
        else {
            break;
        }
    }
    inputLast[length - i - 1] = '\0';

    long inputFirstLong = atoi(inputFirst);
    long inputLastLong = atoi(inputLast);
    int change =0;

```



```

        if(inputFirstLong < inputLastLong){
            long temp;
            temp = inputFirstLong;
            inputFirstLong = inputLastLong;
            inputLastLong = temp;
            change=1;
        }

        long outputValueLong = syscall(451, inputFirstLong, inputLastLong, operators);
        if (operators==0 && change ==1){
            outputValueLong =0- outputValueLong;
        }
        printf("Output: %ld\n", outputValueLong);
    }
    else{
        printf("Wrong Input!\n");
    }
}
return 0;
}

```

마. assignment2.c 추가설명

기본적으로 루프를 도는데, 루프 중단 방법은 NULL 값이 입력값일 때, \n만 입력받았을 때, 중단한다. 입력받은 메시지에서 '+'와 '-'가 포함되는지 검사하고 해당 부호가 포함된다면 program 변수의 값을 2로 설정하고, +인 경우엔 operators의 값을 0으로, -인 경우엔 operators의 값을 1로 설정한다.

만약 부호를 만나지 않는다면 program을 1로 설정한다. 여기서 숫자와 덧셈, 뺄셈 이외의 다른 문자가 포함된다면 program을 0으로 설정한다.

이제 program 값에 따라서 실행 방향이 바뀌는데, program이 1인 경우엔 숫자로만 이루어진 경우로 자릿수 역순의 값을 출력해야한다. 따라서 받은 값을 String에서 long으로 바꾼 뒤, sys_print_reverse 시스템콜의 변수로 보내고 리턴값을 출력한다.

program이 2인 경우엔 숫자, 부호, 숫자로 입력된 경우로 앞부분의 숫자와 뒷부분의 숫자를 따로 분리한다. 그 전에 우선 이때 분리하는 배열을 하나씩 루프를 돌면서 읽은 뒤 또 다른 배열로 복사하고 이 과정을 루프가 부호를 만나기 전까지 진행한다. 첫 번째 숫자를 저장한 뒤 부호 다음에 이어지는 숫자로 같은 방식으로 진행한다.

이렇게 얻은 첫 번째 숫자, 두 번째 숫자 배열을 정숫값으로 바꾼 뒤 operators 값과 함께 sys_print_reverse2시스템콜의 변수로 보낸다. 주의할 점은 보낼 때 operators가 0일 땐 시스템콜에선 뺄셈으로 작동하므로 첫 번째 숫자와 두 번째 숫자의 크기를 비교해서 큰 수를 첫 번째 숫자로 보내고 작은 숫자를 두 번째 숫자로 정한 뒤 시스템콜의 변수로 보낸다.

반환받은 값을 출력한다.

program의 값이 0인 경우엔 Wrong Input!을 출력한다.