OS Project 1

Operating System, Spring 2020

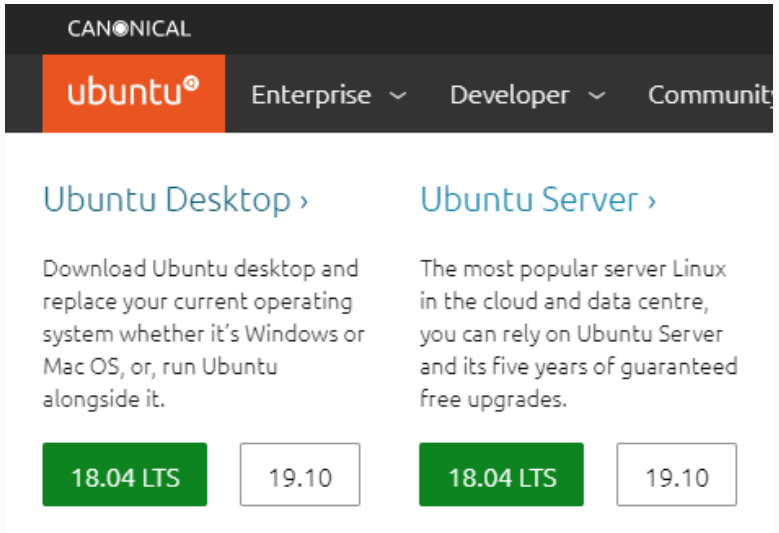
Assignment 0:

Compiling Linux kernel + Adding a simple system call

1 Install Ubuntu and VirtualBox

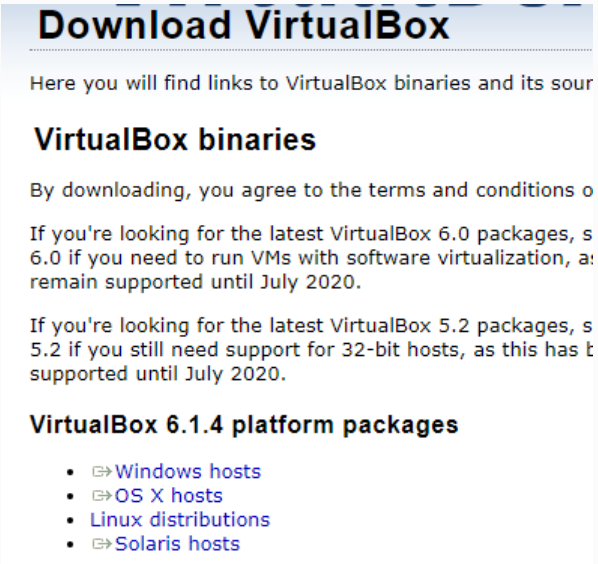
1. Ubuntu Desktop 18.04 LTS Download

https://ubuntu.com/#download



2. VirtualBox Download and Install

https://www.virtualbox.org/wiki/Downloads



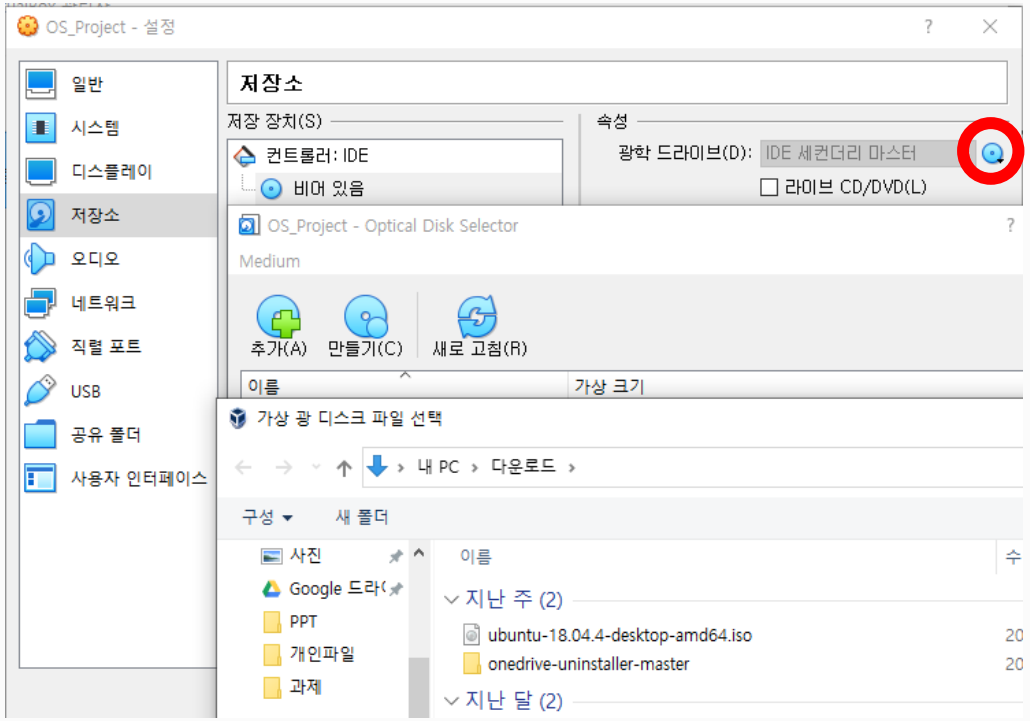
1 Install Ubuntu and VirtualBox

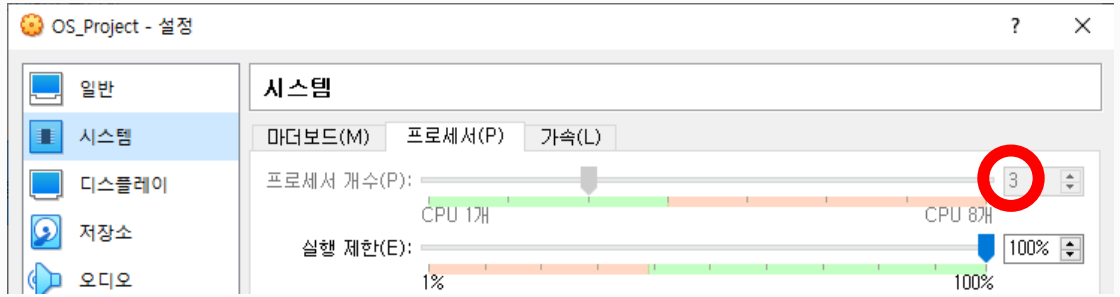
3. Virtual Box Configuration

Memory : roughly 4096MB
Virtual Hard disk(VDI) : at least 50GB

해당 가상환경의 설정에서
저장소-컨트롤러:IDE-비어있음-광학드라이브 Click
다운받은 ubuntu-18.04 선택 후 확인

Set the number of processor to be 1 less than the number of physical cores of the PC CPU





1 Install Ubuntu and VirtualBox

4. Boot Ubuntu

5. Modify grub Configuration

$ sudo vi /etc/default/grub

Adding # in the following two lines

GRUB_TIMEOUT_STYLE=hidden

GRUB_TIMEOUOT=0

```
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_TIMEOUT_STYLE=hidden
#GRUB_TIMEOUT=0
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"
```

$ sudo update-grub

```
ryotta205@ryotta205-VirtualBox:~$ sudo vi /etc/default/grub
ryotta205@ryotta205-VirtualBox:~$ sudo update-grub
Sourcing file `/etc/default/grub'
grub 설정 파일을 형성합니다 ...
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
완료되었습니다
```

Upon rebooting the following screen is displayed.

```
GNU GRUB  버전 2.02

*Ubuntu
 Ubuntu용 고급 설정
 Memory test (memtest86+)
 Memory test (memtest86+, serial console 115200)
```

2 Install new Linux kernel

1. Check the current kernel version and download the 5.5.16 version kernel.

$ uname –r

```
ryotta205@ryotta205-VirtualBox:~$ uname -r
5.3.0-28-generic
```

2. Download new kernel (kernel version should be 5.5.16)

$wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.5.16.tar.xz

or

https://www.kernel.org/

mainline:	5.6	2020-03-29	[tarball]	[pgp]	[patch]
stable:	5.6.3	2020-04-08	[tarball]	[pgp]	[patch] [inc. patch]
stable:	5.5.16	2020-04-08	[tarball]	[pgp]	[patch] [inc. patch]
longterm:	5.4.31	2020-04-08	[tarball]	[pgp]	[patch] [inc. patch]
longterm:	4.19.114	2020-04-02	[tarball]	[pgp]	[patch] [inc. patch]

3. Kernel decompression and Package download

$ sudo mv linux-5.5.16.tar.xz /usr/src/
$ cd /usr/src
$ sudo xz -d linux-5.5.16.tar.xz
$ sudo tar xf linux-5.5.16.tar
$ cd linux-5.5.16
$ ls

```
ryotta205@ryotta205-VirtualBox:/usr/src$ sudo tar xf linux-5.5.16.tar
ryotta205@ryotta205-VirtualBox:/usr/src$ cd linux-5.5.16
ryotta205@ryotta205-VirtualBox:/usr/src/linux-5.5.16$ ls
COPYING        Kconfig      README  crypto   init    mm       security  virt
CREDITS        LICENSES     arch    drivers  ipc     net      sound
Documentation  MAINTAINERS  block   fs       kernel  samples  tools
Kbuild         Makefile     certs   include  lib     scripts  usr
ryotta205@ryotta205-VirtualBox:/usr/src/linux-5.5.16$
```

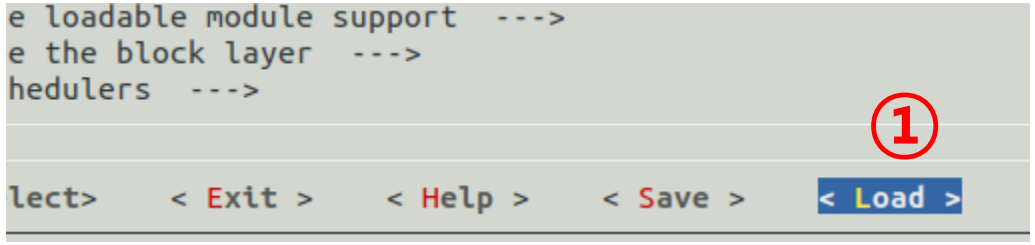
$ sudo apt update
$ sudo apt install build-essential libncurses5 libncurses5-dev bin86 kernel-package libssl-dev bison flex libelf-dev

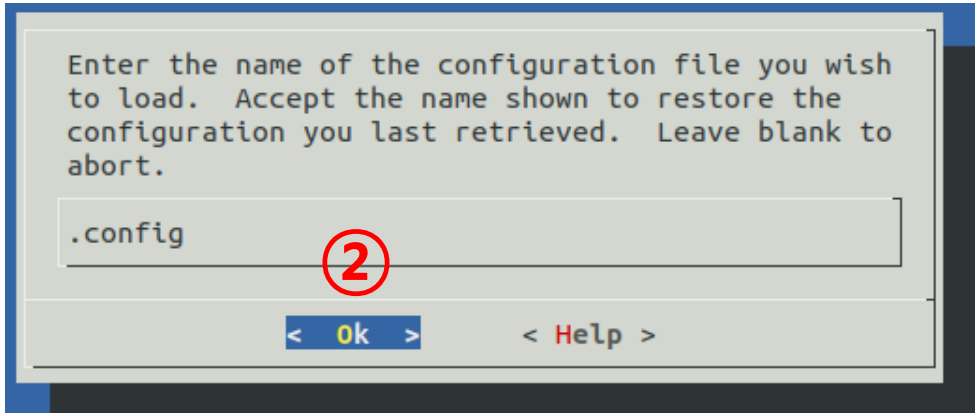
2 Install new Linux kernel - Compile

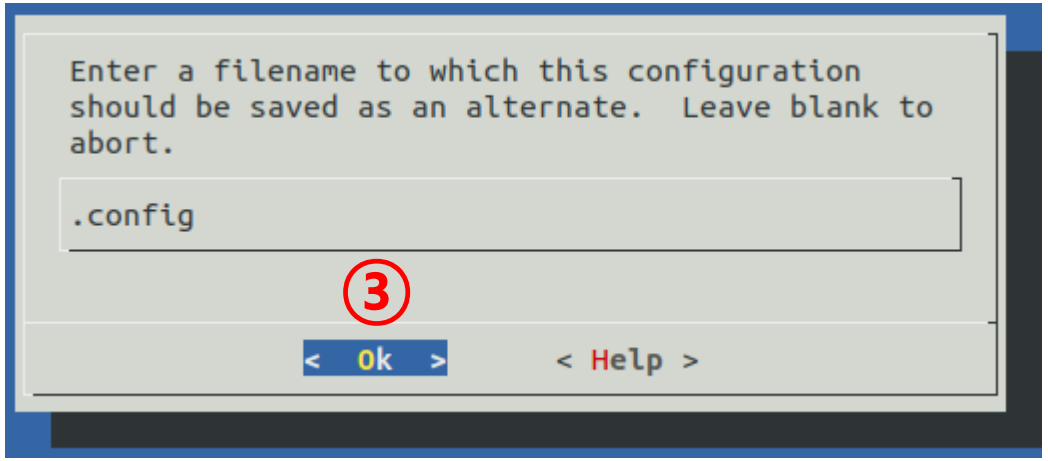
4. Use the current kernel configuration as it is.

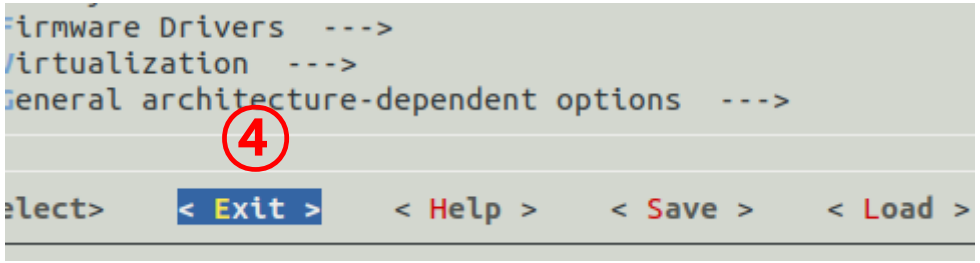
```
$ cd /usr/src/linux-5.5.16
$ sudo cp /boot/config-5.3.0-28-generic ./.config
$ sudo make menuconfig
```

Using the arrow keys
Load – Ok – Save – Ok - Exit









2 Install new Linux kernel - Compile

5. Compile and Install

Compile and install the downloaded kernel and kernel module

$ sudo make –j3

Number of after j = number of allocated CPU cores

$ grep –c processor /proc/cpuinfo Use this to check the number of allocated CPU cores

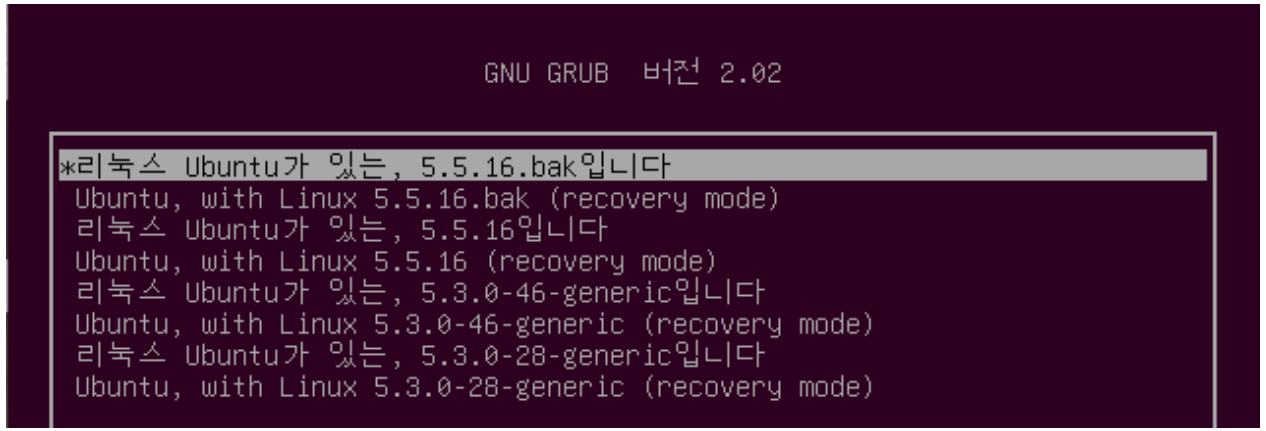
$ sudo make modules_install

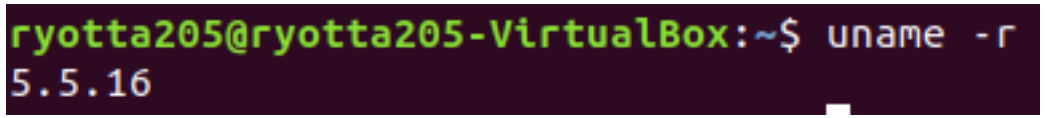
$ sudo make install

$ sudo reboot

Depending on performance, 30 minutes ~ 5 hours is required

Upon rebooting the following screen is displayed. (Ubuntu용 고급 설정 선택)





3 Adding a System Call

1. Make a simple System call

```
$ cd /usr/src/linux-5.5.16
$ sudo vi kernel/hello.c
```

hello.c

```c
#include <linux/kernel.h>
#include <linux/linkage.h>
#include <linux/syscalls.h>
int sys_hello(int x, int y){
        printk("x : %d",x);
        printk("y : %d",y);
        printk("HELLO_SYSTEMCALL_x*y = %d\n",x*y);
        return x*y;
}

SYSCALL_DEFINE2(hello, int, x, int, y)
{
        return sys_hello(x,y);
}
```

2. Register a System call in System call table

```
$ cd /usr/src/linux-5.5.16
$ sudo vi arch/x86/entry/syscalls/syscall_64.tbl
436        common   hello        __x64_sys_hello
```

syscall_64.tbl

```
433     common  fspick                  __x64_sys_fspick
434     common  pidfd_open              __x64_sys_pidfd_open
435     common  clone3                  __x64_sys_clone3/ptregs
436     common  hello                           __x64_sys_hello

#
# x32-specific system call numbers start at 512 to avoid cache impact
```

3 Adding a System Call

3. Declaration a System call prototype

$ cd /usr/src/linux-5.5.16

$ sudo vi include/linux/syscalls.h

Register under '#ifndef CONFIG_ARCH_HAS_SYS_CALL_WRAPPER'

asmlinkage int sys_hello(int x, int y);

```
 * for architectures overriding the syscall calling convention, do not
 * include the prototypes if CONFIG_ARCH_HAS_SYSCALL_WRAPPER is enabled.
 */
#ifndef CONFIG_ARCH_HAS_SYSCALL_WRAPPER
asmlinkage int sys_hello(int x, int y);
asmlinkage long sys_io_setup(unsigned nr_reqs, aio_context_t __user *ctx);
asmlinkage long sys_io_destroy(aio_context_t ctx);
```

4. Modify Makefile

$ cd /usr/src/linux-5.5.16

$ sudo vi kernel/Makefile

Add 'hello.o' to the end of 'oby-y'

```
obj-y     = fork.o exec_domain.o panic.o \
            cpu.o exit.o softirq.o resource.o \
            sysctl.o sysctl_binary.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 hello.o
```

3 Adding a System Call

5. Recompile

Compile only kernel image and exchange (5~10 minute)

$ cd /usr/src/linux-5.5.16
$ sudo make bzImage –j3
(Number of after j = number of allocated CPU cores)
$ sudo cp arch/x86/boot/bzImage /boot/vmlinuz-5.5.16
$ reboot

```
root@ryotta205-VirtualBox:/usr/src/linux-5.5.16# make bzImage -j3
  DESCEND  objtool
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
  CC      init/main.o
```

```
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Setup is 17724 bytes (padded to 17920 bytes).
System is 9157 kB
CRC ca10ee9b
Kernel: arch/x86/boot/bzImage is ready  (#9)
root@ryotta205-VirtualBox:/usr/src/linux-5.5.16# sudo cp arch/x86/boot/bzImage /boot/vmlinuz-
5.5.16
```

3 Adding a System Call

6. Test

$ vi test.c
$ gcc test.c
$ ./a.out
$ dmesg

test.c

```c
#include <unistd.h>
#include <stdio.h>

#define __NR_hello 436

int main(void){

        int n=0;
        n = syscall(__NR_hello, 9, 7);
        printf("return value : %d\n", n);
        return 0;
}
```

```
ryotta205@ryotta205-VirtualBox:~$ vi test.c
ryotta205@ryotta205-VirtualBox:~$ gcc test.c
test.c: In function ‘main’:
test.c:9:6: warning: implicit declaration of function ‘syscall’; did you mean ‘s
scanf’? [-Wimplicit-function-declaration]
  n = syscall(__NR_hello, 9, 7);
      ^~~~~~~
      sscanf
ryotta205@ryotta205-VirtualBox:~$ ./a.out
return value : 63
ryotta205@ryotta205-VirtualBox:~$
```

```
[  127.075280] x : 9
[  127.075281] y : 7
[  127.075283] HELLO_SYSTEMCALL_x*y = 63
ryotta205@ryotta205-VirtualBox:~$
```

4 Troubleshooting

1. 만약 부팅 중 Ubuntu 로고에서 오랫동안 멈춰 있다면 재부팅
   여러 번의 재부팅 이후로도 안된다면 2번 진행

2. 만약 부팅 중 Started GNOME Display Manager에서 멈춰 있다면
   Recovery mode 진입 후
   root – Enter
   $ apt-get purge gdm3
   $ apt-get install gdm3 ubuntu-desktop
   $ reboot

```
         Starting Hostname Service...
[  OK  ] Started Permit User Sessions.
         Starting GNOME Display Manager
         Starting Hold until boot proce
[  OK  ] Started Authorization Manager.
[  OK  ] Started Accounts Service.
[  OK  ] Started Hostname Service.
         Starting Network Manager Script
[  OK  ] Started Network Manager Script
[  OK  ] Started The PHP 7.2 FastCGI Pro
[  OK  ] Started Disk Manager.
[  OK  ] Started Clean php session files
[  OK  ] Started GNOME Display Manager.
```

```
                    GNU GRUB  버전 2.02

 리눅스 Ubuntu가 있는, 5.5.16입니다
*Ubuntu, with Linux 5.5.16 (recovery mode)
 리눅스 Ubuntu가 있는, 5.3.0-46-generic입니다
```

```
root@ryotta205-VirtualBox:~# apt-get purge gdm3
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  gdm3* ubuntu-desktop*
```

```
root@ryotta205-VirtualBox:~# apt-get install gdm3 ubuntu-desktop
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  gnome-orca libpam-fprintd
The following NEW packages will be installed:
  gdm3 ubuntu-desktop
```

Assignment 1:

Adding “sys_mygetsid()“ system call

1. Add sys_mygetsid()

- **sys_mygetsid(int* buffer)**
  - 매개변수로 전달한 포인터 변수에 자신의 학번을 전달하는 system call

Test user 프로그램 소스코드: sys_mygetsid()

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

#define __NR_hello 436
#define __NR_mygetsid 437

int main(void){

        int *n;

        n = (int*)calloc(1,sizeof(int));
        printf("before buffer value : %d\n", *n);
        syscall(__NR_mygetsid, n);
        printf("after buffer value : %d\n", *n);
        return 0;
}
```

실행결과

```
ryotta205@ryotta205-VirtualBox:~$ ./a.out
before buffer value : 0
after buffer value : 2015114398
ryotta205@ryotta205-VirtualBox:~$
```

dmesg 확인결과

```
[ 1157.676032] system call : sys_mygetsid()
[ 1157.676033] system call : sys_mygetsid() : 2015114398
```

2. 제출 및 평가

제출 자료 (lms에 제출)

1. Linux kernel image (vmlinuz) file
2. source code of the system call
3. source code of the test user program
4. Result of execution (Screenshot)

평가

1. sys_hello() System call 추가 (20점)
2. System call 추가 후 Compile 성공 (10점)
3. Compile된 커널로 부팅 성공 (10점)
4. sys_mygetsid() 구현 (20점)

Assignment 2:

Adding “sys_sread()”, “sys_swrite()“ system call

1. Add sys_swrite(), sys_sread()

- **sys_swrite(int fd, char* buf, int len): A secure version of sys_write() system call**
  - buf 포인터 변수가 가리키는 메모리 공간에 저장된 데이터를 암호화(encryption)한 뒤, fd가 가리키는 파일에 len 크기 만큼 저장하는 system call
  - Encryption 방법: 데이터의 모든 비트를 반전 시킴
    예) 원본 데이터 (01110000) → 암호화된 데이터 (10001111)

- **sys_sread(int fd, char* buf, int len): A secure version of sys_read() system call**
  - fd가 가리키는 파일에서 len 크기만큼 읽어 복호화(decryption)한 뒤, buf포인터 변수가 가리키는 메모리 공간에 저장하는 system call
  - 파일에서 읽은 데이터는 복호화한 뒤 buf에 저장해야 함

1. Add sys_swrite(), sys_sread()

Test user 프로그램 소스코드: sys_swrite()

```c
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

#define __NR_hello 436
#define __NR_mygetsid 437
#define __NR_swrite 438
#define __NR_sread 439

#define BUFSIZE 100

void to_binary(char* buf){
        unsigned char p;

        for(int j = 0; j <strlen(buf) ; j++){
                p = 0x80;
                for(int i=0; i<8 ;i++){
                        if(buf[j] & p){
                                printf("1");
                        }else{
                                printf("0");
                        }
                        p = p >> 1;
                }
                printf(" ");
        }
        printf("\n");
}
```

```c
int main()
{
        char  *temp1;
        char  *temp2;
        int    fd;

        temp1 = (char*)calloc(100,sizeof(char));
        temp2 = (char*)calloc(100,sizeof(char));

        printf("Input data : ");
        scanf("%s",temp1);
        printf("Input binary data : ");
        to_binary(temp1);

        if ( 0 < ( fd = open( "./test.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644))){
                syscall(__NR_swrite, fd, temp1, strlen(temp1));
                close(fd);
        }
        else{
                printf( "Open Error\n");
                exit(-1);
        }

        if ( 0 < ( fd = open( "./test.txt", O_RDONLY, 0644))){
                read(fd, temp2, BUFSIZE);
                close(fd);
        }
        else{
                printf( "Open Error.\n");
                exit(-1);
        }

        printf("sys_read() data : %s\n",temp2);
        printf("sys_read() binary data : ");
        to_binary(temp2);

        return 0;
}
```

1. Add sys_swrite(), sys_sread()

Test user 프로그램 소스코드: sys_sread()

```c
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

#define __NR_hello 436
#define __NR_mygetsid 437
#define __NR_swrite 438
#define __NR_sread 439

#define BUFSIZE 100

void to_binary(char* buf){
        unsigned char p;

        for(int j = 0; j <strlen(buf) ; j++){
                p = 0x80;
                for(int i=0; i<8 ;i++){
                        if(buf[j] & p){
                                printf("1");
                        }else{
                                printf("0");
                        }
                        p = p >> 1;
                }
                printf(" ");
        }
        printf("\n");
}
```

```c
int main()
{
        char  *temp2;
        char  *temp3;
        int    fd;

        temp2 = (char*)calloc(100,sizeof(char));
        temp3 = (char*)calloc(100,sizeof(char));

        if ( 0 < ( fd = open( "./test.txt", O_RDONLY, 0644))){
                read(fd, temp2, BUFSIZE);
                close(fd);
        }
        else{
                printf( "Open Error.\n");
                exit(-1);
        }

        printf("sys_read() data : %s\n",temp2);
        printf("sys_read() binary data : ");
        to_binary(temp2);

        if ( 0 < ( fd = open( "./test.txt", O_RDONLY, 0644))){
                syscall(__NR_sread, fd, temp3, BUFSIZE);
                close(fd);
        }
        else{
                printf( "Open Error.\n");
                exit(-1);
        }

        printf("sys_sread() data : %s\n",temp3);
        printf("sys_sread() binary data : ");
        to_binary(temp3);

        return 0;
}
```

1. Add sys_swrite(), sys_sread()

Test user 프로그램 소스코드: sys_sread() 실행결과

```
ryotta205@ryotta205-VirtualBox:~$ ./a.out
Input data : Linux
Input binary data : 01001100 01101001 01101110 01110101 01111000
sys_read() data : ◆◆◆◆◆
sys_read() binary data : 10110011 10010110 10010001 10001010 10000111
```

dmesg 확인결과

```
[  844.154938] sys_swrite()
```

Test user 프로그램 소스코드: sys_sread() 실행결과

```
ryotta205@ryotta205-VirtualBox:~$ ./a.out
sys_read() data : ◆◆◆◆◆
sys_read() binary data : 10110011 10010110 10010001 10001010 10000111
sys_sread() data : Linux
sys_sread() binary data : 01001100 01101001 01101110 01110101 01111000
```

dmesg 확인결과

```
[  940.868545] sys_sread()
```

2. Make a library for sys_swrite(), sys_sread()

How to make a library?

Static Library

```
$ vi hello.c
$ gcc –c hello.c
$ ar rc libhello.a hello.o
$ vi static.c
$ gcc static.c –o static –L./ -lhello
$ ./static
```

hello.c

```c
#include <unistd.h>

#define __NR_hello 436

int hello(int n, int m){
        return syscall(__NR_hello, n, m);
}
```

static.c

```c
#include <unistd.h>
#include <stdio.h>

int main(){
        int n;

        n = hello(9,9);
        printf("hello return : %d\n", n);

        return 1;
}
```

결과

```
ryotta205@ryotta205-VirtualBox:~$ ./static
hello return : 81
```

2. Make a library for sys_swrite(), sys_sread()

How to make a library?

Shared Library

```
$ vi hello.c
$ gcc –fPIC –c hello.c –o hello.o
$ gcc –shared –o libhello.so.0.0.0 hello.o
$ sudo cp libhello.so.0.0.0 /usr/lib/libhello.so.0.0.0
$ sudo ln –s /usr/lib/libhello.so.0.0.0 /usr/lib/libhello.so
$ sudo vi /etc/ld.so.conf.d/libhello.conf
$ sudo ldconfig
$ vi shared.c
$ gcc shared.c –o shared -lhello
$ ./shared
$ ldd shared
```

ldd shared

```
ryotta205@ryotta205-VirtualBox:~$ ldd shared
        linux-vdso.so.1 (0x00007fffa2137000)
        libhello.so => /usr/lib/libhello.so (0x00007f8baa2dc000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f8ba9eeb000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f8baa6e0000)
```

hello.c

```c
#include <unistd.h>

#define __NR_hello 436

int hello(int n, int m){
        return syscall(__NR_hello, n, m);
}
```

shared.c

```c
#include <unistd.h>
#include <stdio.h>

int main(){
        int n;

        n = hello(9,9);
        printf("hello return : %d\n", n);

        return 1;
}
```

libhello.conf

```
# libc default configuration
/usr/lib
```

결과

```
ryotta205@ryotta205-VirtualBox:~$ ./static
hello return : 81
```

2. Make a library for sys_swrite(), sys_sread()

- **Make a library providing user-level wrapper functions for sys_swrite() and sys_sread() system calls.**

- **Library function name should be sread() and swrite() for sys_sread() and sys_swrite(), respectively**

3. 제출 및 평가

제출 자료 (lms에 제출)

1. Linux kernel image (vmlinuz) file
2. source code of the system call
3. source code of the test user program
4. source code of the library
5. Result of execution (Screenshot)

평가

1. sys_sread() system call 추가(5점)
2. sys_swrite() system call 추가 (5점)
3. sys_sread() 구현 (10점)
4. sys_swrite() 구현 (10점)
5. Library 구현 (10점)

Question

Email : jhyn207@gmail.com
Message : 010 - 4905 - 1749