System Call 구현

- Linux 커널의 새로운 시스템 호출 구현은 아래 두 가지 작업으로 구성
  - » 커널 수정
  - » 새로운 시스템 호출을 이용하는 사용자 수준 응용
- 세부단계
  - » 커널 수정
    - 시스템 호출 번호 할당
    - 시스템 호출 테이블 등록
    - 시스템 호출 처리 함수 구현
    - 커널 컴파일 및 리부팅
  - » 사용자 수준 응용 작성
    - 시스템 호출을 사용하는 프로그램 작성
    - 라이브러리 작성(꼭 필요한 것은 아님)
- 커널을 수정하기 위해서는 커널 소스 필요
  - » RedHat 9.0의 경우 커널 개발 패키지를 설치하면 /usr/src 디렉토리 밑에 설치됨 (커널 버전 2.4.20-8)
  - » 커널 소스를 구해서 /usr/src 디렉토리 밑에 설치하고 설치한 소스 디렉토리를 linux로 심볼릭 링크를 건다.
  - » ln –s /usr/src/linux-2.4.20-8 /usr/src/mylinux

커널 수정

- 시스템 호출 번호 할당
  - » Linux 커널이 제공하는 시스템 호출은 각각 고유한 번호를 갖는다. (Redhat 9.0의 커널(2.4.20-8)은 0~259번까지 지원)
  - » /usr/src/mylinux/include/asm-i386/unistd.h 파일에 구현되어 있다.

```
/* include/asm-i386/unistd.h 파일의 내용 *
#ifndef _ASM_I386_UNISTD_H_
#define _ASM_I386_UNISTD_H_

#define __NR_exit              1
#define __NR_fork              2
#define __NR_read              3
…
#define __NR_lookup_dcookie            253
#define __NR_set_tid_address           258
```

  - » #define __NR_newsyscall 259 를 추가 한다.
  - » /usr/include/asm/unistd.h 파일도 똑같이 수정해 준다.

커널 수정 (2)

- 시스템 호출 테이블 등록
  - » 시스템 호출처리 함수는 sys_call_table이라는 테이블에 등록이 되어 있어야 한다
  - » sys_call_table 정보는 /usr/src/mylinux/arch/i386/kernel/entry.S 파일에 구현되어 있다
  - » /*258*/ 다음 라인에 .long SYMBOL_NAME(sys_newsyscall) 라인을 추가한다

```
.data
ENTRY(sys_call_table)
        .long SYMBOL_NAME(sys_ni_syscall)       /* 0  -  old "setup()" system call*/
        .long SYMBOL_NAME(sys_exit)
        .long SYMBOL_NAME(sys_fork)
        .long SYMBOL_NAME(sys_read)
…
         .long SYMBOL_NAME(set_tid_address)              /*258*/

         .long SYMBOL_NAME(sys_newsyscall)               /*259*/

         .rept NR_syscalls-(.-sys_call_table)/4
                        .long SYMBOL_NAME(sys_ni_syscall)
        .endr
```

커널 수정 (3)

- 시스템 호출 처리 함수 구현
  - » 일반 적으로 태스크 관리자 관련 함수는 /usr/src/mylinux/kernel/ 파일 시스템 관련 함수는 /usr/src/mylinux/fs/ 밑에 구현 되어 있다.
  - » 구현할 함수를 /usr/src/mylinux/kernel/ 디렉토리 밑에 newfile.c라는 이름으로 구현한다.

```
/* /usr/src/mylinux/kernel/newfile.c */
#include <linux/linkage.h>
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/sched.h>

asmlinkage int sys_newsyscall()
{
  printk("Hello Linux, I'm in Kernel\n");
      return(0);
}
```

  - » /usr/src/mylinux/kernel/Makefile 수정
    - obj -y = 라인에 newfile.o 추가

커널 수정 (4)

- 커널 생성
  [$ rm -rf /usr/src/mylinux (이전 심볼릭 링크 삭제)] 생략 가능
  $ ln -s /usr/src/linux-2.4.20-8 /usr/src/mylinux
  $ cd /usr/src/mylinux (커널 소스의 상위 디렉토리로 이동)
  $ make mrproper (기존의 의존성 제거)
  $ make menuconfig (범용적 커널 설정 스크립트 실행)

- 커널 컴파일
  » 옵션은 아래 사이트 참고
    - http://doc.kldp.org/KoreanDoc/html/Kernel24_Intro-KLDP/Kernel24_Intro-KLDP-2.html
  $ make dep (커널 소스 구성)
  $ make clean (구 버전 소스 파일 삭제)
  $ make bzImage (컴파일, 새 커널 생성)
- 커널 인스톨
  » 모듈로 구성된 커널 내부 구성 요소를 알려주고, 이후 구성 요소들이 사용될 때 자동으로 커널에 적재되게 함
  $ make modules (모듈 컴파일)
  $ make modules_install (모듈 인스톨)
- 생성된 커널을 /boot 디렉토리에 복사
  $ cp arch/i386/boot/bzImage /boot/mybzImage
  $ cp System.map /boot/System.map-2.4.20-8 (새 System.map으로 대치)
- 부트로더 설정

커널 수정 (5)

- 부트로더 설정 (GRUB)
  - » Vi 편집기로 /boot/grub/grub.conf (또는 menu.lst) 파일 수정

    title RedHat9.0 (add syscall)
    - root(hd0,1) (/boot 파티션이나 커널 이미지가 있는 파티션, 0번부터)
    - kernel /mybzImage ro root=/dev/hda3 (/ 파티션 위치)

      (boot 파티션을 따로 생성했을 경우. 따로 boot 파티션을 생성하지 않았을 때는 커널 이미지 경로를 /boot/로 해준다)

  - » 위 라인 추가 후 다음 명령어 실행

  $ grub-install /dev/hda (생략 해도 됨)

  $ reboot

  - » 부트로더에서 새로 컴파일한 커널 선택 후 부팅

사용자 수준 응용 작성

- 시스템 호출 사용 프로그램 작성

```
#include <linux/unistd.h>
#include <errno.h>
_syscall0(int, newsyscall);

main()
{
int i;
        i=newsyscall();
}
```

- 라이브러리 작성(선택사항)

```
$ gcc –c newsys.c
$ ar –r libnew.a newsys.o
$ ranlib libnew.a
$ vi test.c
$ gcc test.c –L /root(libnew.a가 있는 디렉토리)/ -lnew
$ a.out
```

/* contents of newsys.c */

```
#include <linux/unistd.h>
#include <errno.h>
_syscall0(int, newsyscall);
```

/* contents of test.c */

```
main()
{
        int i;
        i = newsyscall();
}
```

실습 1

- 부교재 p65 sys_gettaskinfo() 라는 새로운 시스템 호출 구현
  - » current (/usr/include/asm/current.h)라는 전역 변수가 task_struct (/usr/src/mylinux/include/linux/sched.h) 자료구조를 포인팅

```
#include <linux/linkage.h>
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/sched.h>

asmlinkage int sys_gettaskinfo()
{
        int i, cnt = 0;

        printk("PID: %d\n", current->pid);
        printk("PPID: %d\n", current->parent->pid);
        if(current->state == -1)        printk("Unrunable state\n");
        else if(current->state == 0)    printk("Runable state\n");
        else if(current->state == 1)    printk("Interruptable state\n");
        else if(current->state == 2)    printk("Uninterruptable state\n");
        else if(current->state == 4)    printk("Stopped state\n");
```

실습 1

```
    else if(current->state == 8)    printk("Zombie state\n");
    else if(current->state == 16)   printk("Dead state\n");
    else                            printk("Unknown state\n");

    printk("Priority: %lu\n", current->rt_priority);
    printk("Scheduling Policy: %lu\n", current->policy);

    printk("User CPU time: %lu ticks\n", current->times.tms_utime);
    printk("System CPU time: %lu ticks\n", current->times.tms_utime);
    printk("Start time: %lu\n", current->start_time);

    printk("Number of major faults: %lu\n", current->maj_flt);
    printk("Number of minor faults: %lu\n", current->min_flt);

    for(i=0; i<256; i++)
        if(current->files->fd_array[i] != NULL)
            cnt++;
    printk("Number of opened file: %d\n", cnt);
    return(0);
}
```

실습 1

- 사용자 응용 작성

```
#include <linux/unistd.h>
#include <errno.h>
_syscall0(int, gettaskinfo);

main()
{
        int i;
        i=gettaskinfo();
}
```

실습 2

- 부교재 P70 sys_getstat(int id, struct mystat *user_buf) 구현
- **/* 헤더 파일 mystat.h */**

```
struct mystat {
  int pid;
  int ppid;
  /*
   * pid_t pid;
   * pid_t ppid;
   */

  int state;
  int priority;
  int policy;
  long utime;
  long stime;
  long starttime;
  unsigned long min_flt;
  unsigned long maj_flt;
  int open_files;};
```

실습 2

```
/* getstat.c */
#include <linux/unistd.h>
#include <linux/errno.h>
#include <linux/sched.h>
#include <asm-i386/uaccess.h>
#include "mystat.h"
//  #include <linux/malloc.h>
#include <linux/slab.h>
asmlinkage int sys_getstat(int id, struct mystat *user_buf)
{
  struct mystat *buf;
            int i = 0, cnt = 0;
  struct task_struct *search;
  search = &init_task;

  while(search->pid != id)
  {
      search = next_task(search);
      if(search->pid == init_task.pid)
          return(-1);
           }
```

실습 2

```
    buf = kmalloc(sizeof(struct mystat), GFP_KERNEL);
    if(buf == NULL)
        return(-1);
    buf->pid = search->pid;
    buf->ppid = search->parent->pid;
    buf->state = search->state;
    buf->priority = search->rt_priority;
    buf->policy = search->policy;
    buf->utime = search->times.tms_utime;
    buf->stime = search->times.tms_stime;
    buf->starttime = search->start_time;
    buf->min_flt = search->min_flt;
    buf->maj_flt = search->maj_flt;
    for(i=0; i<256; i++)
        if(current->files->fd_array[i] != NULL)
            cnt++;
    buf->open_files = cnt;
    copy_to_user(user_buf, buf, sizeof(struct mystat));
    return(0);
}
```

실습 2

- 사용자 응용

```
#include <linux/unistd.h>
#include <stdio.h>
#include <errno.h>
#include "mystat.h“
struct mystat *mybuf;
_syscall2(int, getstat, int, taskid, struct mystat *, ret_buf);

int main(int argc, int* argv[])
{
      int task_number;
      if(argc != 2){
            printf("USAGE: a.out pid\n");
            exit(1);
      }
      task_number = atoi(argv[1]);
      mybuf = (struct mystat *)malloc(sizeof(struct mystat));
      if(mybuf == NULL){
            printf("Out of Memory\n");
            exit(1);
      }
```

실습 2

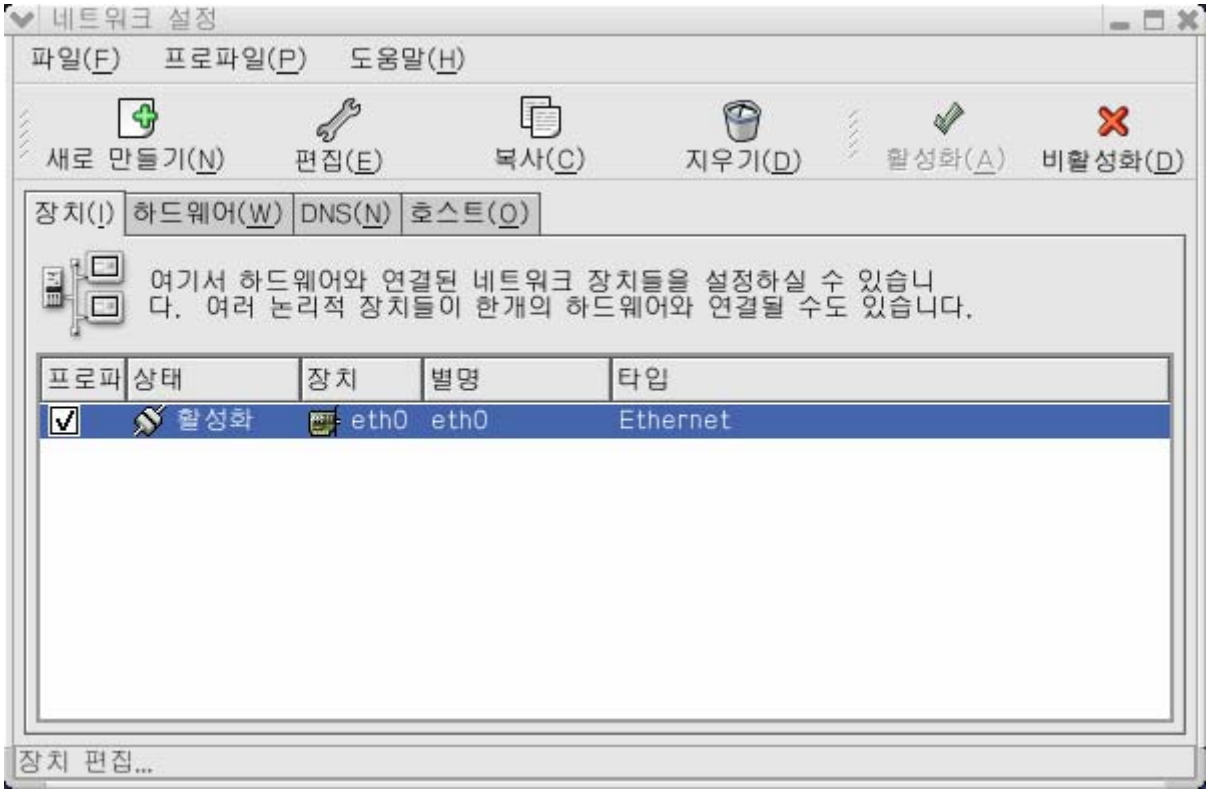
```
 getstat(task_number, mybuf);
printf("PID = %d\n", mybuf->pid);
printf("PPID = %d\n", mybuf->ppid);

if(mybuf->state == -1)          printf("Unrunable state\n");
else if(mybuf->state == 0)      printf("Running state\n");
else if(mybuf->state == 1)      printf("Interruptable state\n");
else if(mybuf->state == 2)      printf("Uninterruptable state\n");
else if(mybuf->state == 4)      printf(" Stopped state\n");
else if(mybuf->state == 8)      printf(" Zombie state\n");
else if(mybuf->state == 16)     printf(“Dead state\n");
else                            printf("Unknown state\n");

printf("Priority = %d\n", mybuf->priority);
printf("Policy = %d\n", mybuf->policy);
printf("Task.utime = %lu\n", mybuf->utime);
printf("Task.stime = %lu\n", mybuf->stime);
printf("Task.starttime = %lu\n", mybuf->starttime);
printf("minor fault = %lu\n", mybuf->min_flt);
printf("major fault = %lu\n", mybuf->maj_flt);
printf("opened files = %lu\n", mybuf->open_files);
}
```

네트워크 설정방법

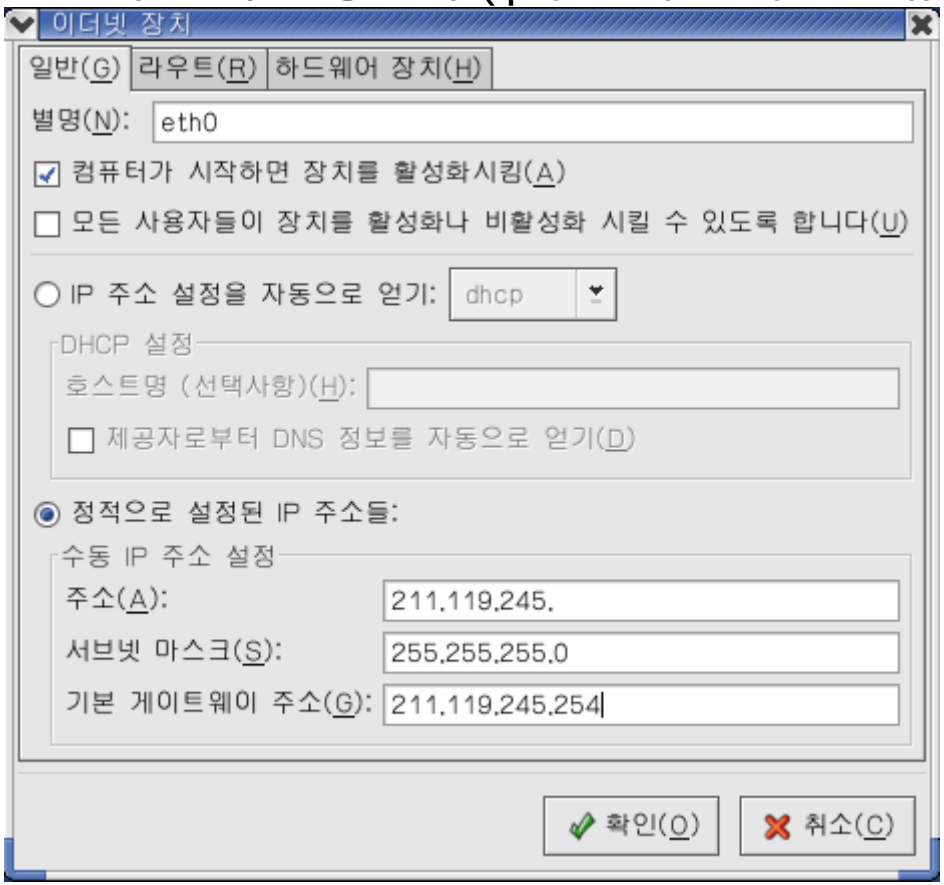
- Redhat-config-network명령을 실행하거나 메뉴에서 시스템 설정에 네트워크를 실행 후 eth0 선택 후 편집을 누른다



- (위와 같이 장치가 잡혀있지 않으면 커널 컴파일 후 모듈설치)

네트워크 설정

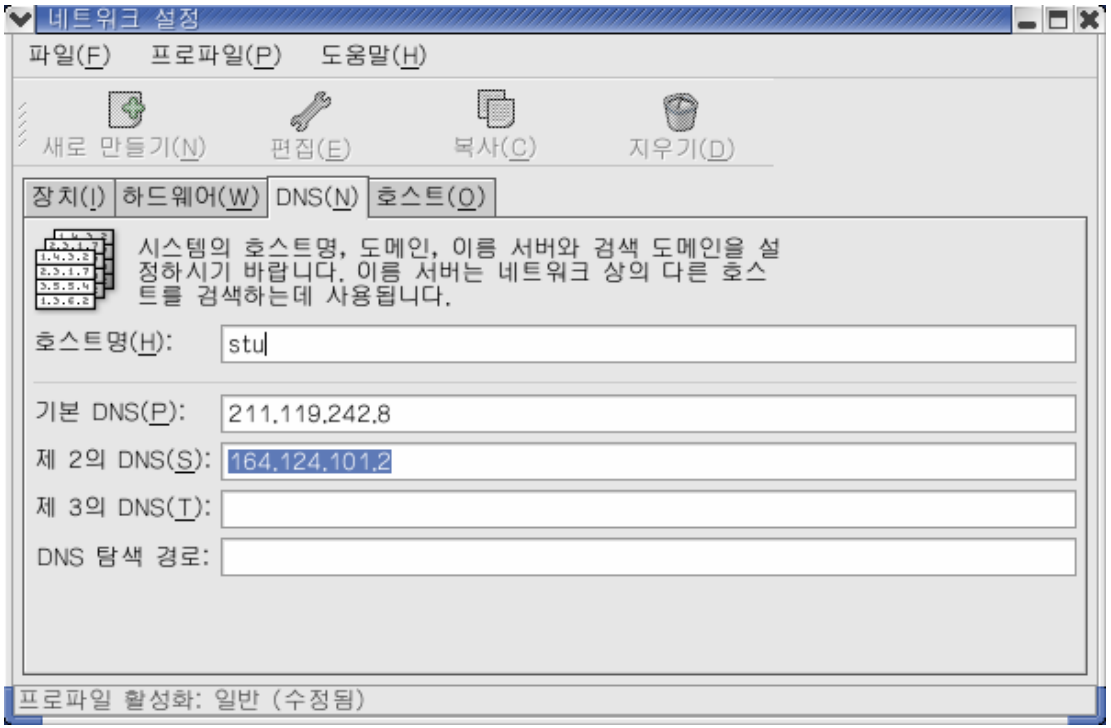
- 그림과 같이 설정한다 (ip주소에는 해당 컴퓨터의 주소 입력)



- 입력이 완료 되면 확인을 누른후 DNS설정을 한다

네트워크 설정

- 호스트명에는 해당 자리의 컴퓨터 이름을 넣는다 (예)stu01



- 설정 완료 후 재부팅하거나 장치를 비활성화 시켰다가 다시 활성화 시킨다
- ping 211.119.245.75 실행 후 응답이 제대로 오면 성공