The file in context

Suntae Hwang
Kookmin University

1 Files in a multi-user environment

- Users and ownerships
  - UNIX의 모든 file은 사용자 중 하나에 소유
  - 소유자의 신원은 user-id (uid)라는 음이 아닌 정수값
  - /etc/passwd

    sthwang:x:500:100:Suntae Hwang:/home/sthwang:/bin/bash
  - /etc/group

- *Effective user- and group-ids*
  - ruid: 실제로 프로세스를 수행시킨 사용자의 uid
  - euid: 생성된 파일에 대해 실제 소유권을 갖는 사용자의 uid
  - ruid, euid 는 프로세스의 속성
  - 한 프로세스가 한 사용자(예: keith)에 의해서 시작되더라도 특수한 경우 다른 사람(예: dina)의 file system 권한을 부여받을 수 있다.

Permissions and file modes

- 파일 소유자는 파일에 연관된 permission을 선택할 수 있다.
- 사용자의 유형
  1. 소유자
  2. 그룹
  3. Others
- 허가의 유형
  1. 읽기
  2. 쓰기
  3. 수행
- Bit pattern으로 File mode를 표현한다.

  -rwxr-xr-x

  0755
- 표 3.1

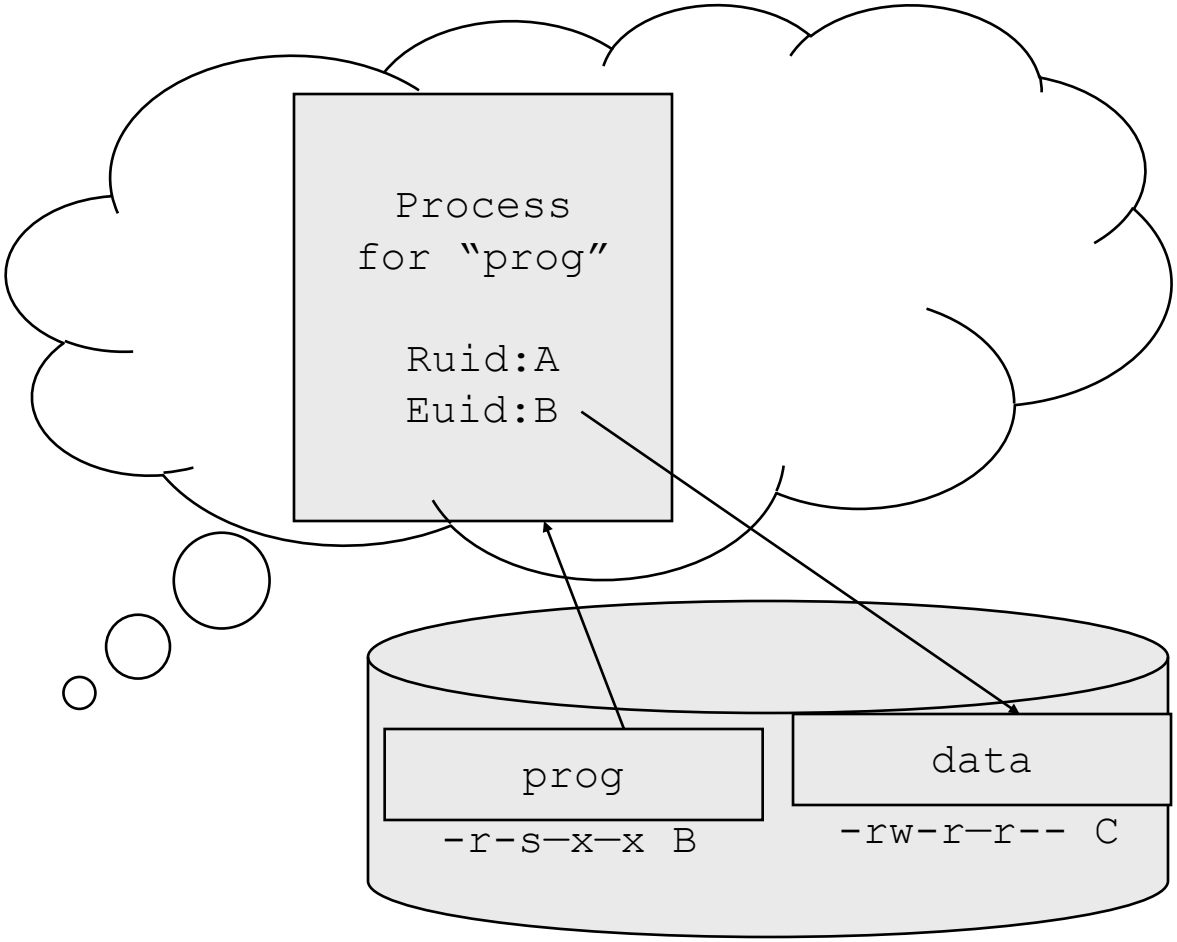
Extra permissions for executable files

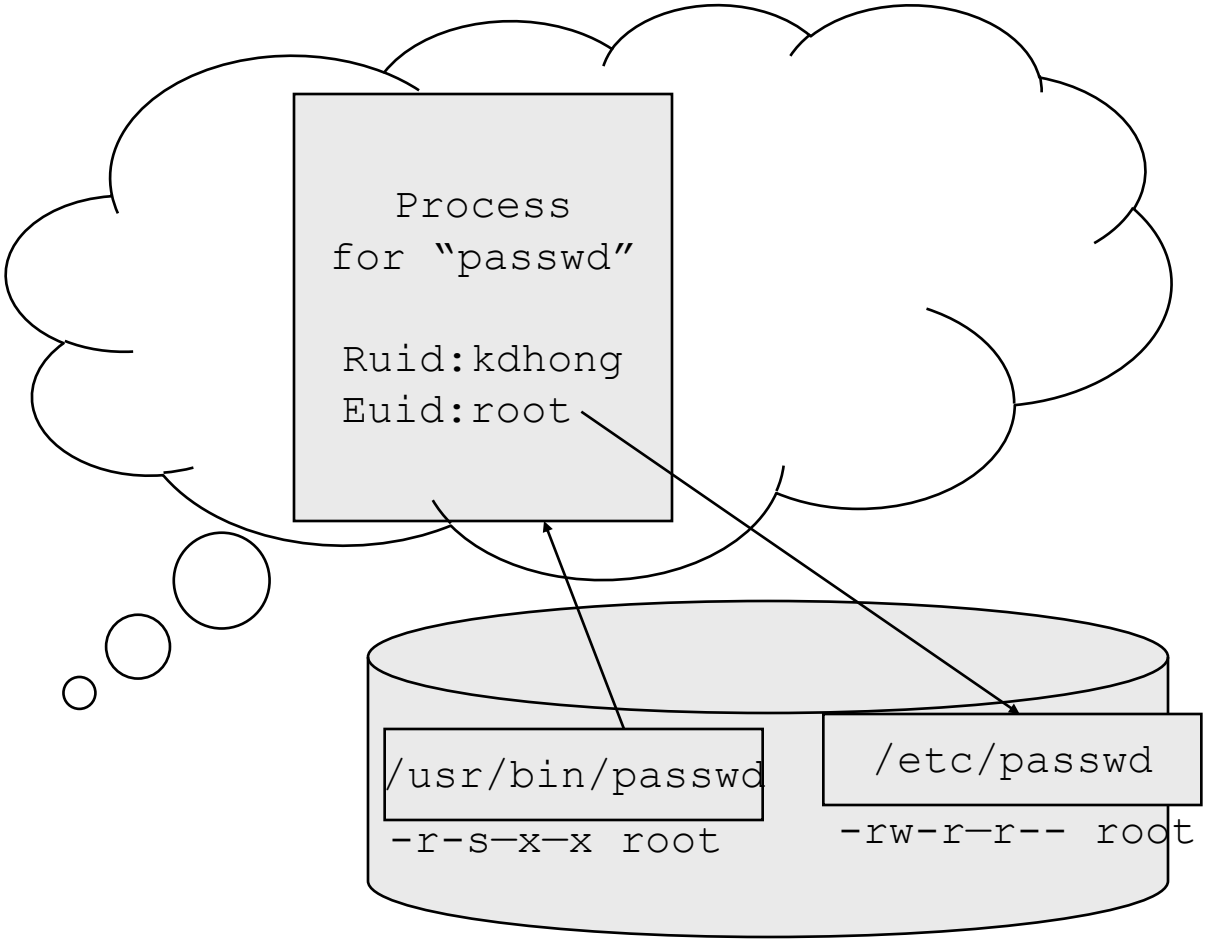
- Three extra permissions

04000	S_ISUID	Set user-id
02000	S_ISGID	Set group-id
01000	S_ISVTX	Sticky bit

  - S_ISUID가 지정되어 있으면, 해당 파일이 수행될 때 파일 소유자의 uid를 생성된 프로세스의 euid로 부여한다.

    예: -rwsr-xr-x
  - passwd라는 명령은 /etc/passwd 또는 /etc/shadow 파일에서 자신의 pass word를 변경하는 것이다. How?
  - Sticky bit: 프로세스 종료 후에도 텍스트 이미지를 swap영역에 남겨 둠, 따라서 다음번 호출할 때 빨리 찾음. 지금은 디렉토리에 대해서만 정의함


Process
for "prog"
Ruid:A
Euid:B
prog
-r-s—x—x B
data
-rw-r—r-- C


Process
for "passwd"
Ruid:kdhong
Euid:root
/usr/bin/passwd
-r-s—x—x root
/etc/passwd
-rw-r—r-- root

The file creation mask

- 파일의 초기 허가는 파일을 생성할 때 설정
- File creation mask: 각 프로세스의 속성

  filedes=open(pathname, O_CREAT, mode);

  Is equivalent to

  filedes=open(pathname, O_CREAT, (~mask) & mode);
- Mask에서 지정된 허가는 항상 0으로 만들어 준다
  - Mask가 07(000000111)이라면

    fd = open("/tmp/newfile", O_CREAT, 0644);

    에서 생성된 파일의 모드는 0640이다.

    ~(000000111) & 0644 = 0640

umask **system call**

```
#include <sys/types.h>
#include <sys/stat.h>
mode_t umask(mode_t newmask);
```

```
mode_t oldmask;
…
oldmask = umask(022);
```

– 파일을 생성할 때 주어진 mask의 영향을 받지 않고 mode대로 생성하려면 …

```
#include <fcntl.h>
#include<sys/stat.h>
int specialcreat(const char *pathname,mode_t mode)
{
   mode_t oldu;
   int filedes;
   if((oldu=umask(0))==-1)
            {
            perror("saving old mask");
            return (-1);
            }
   if((filedes=open(pathname,O_WRONLY|O_CREAT|O_EXCL,mode))==-1)
            perror("opening file");
   if(umask(oldu)==-1)
            perror("restoring old mask");
   return filedes;
}
```

open **and file permissions**

filedes=open(pathname, O_WRONLY | O_CREAT | O_TRUNC, 0600);

- 파일 허가가 호출 프로세스의 쓰기 접근을 허용한다면 파일이 존재할 경우 파일을 잘라 버린다.

filedes=open(pathname, O_WRONLY | O_CREAT | O_EXCL, 0600);

- 어떤 허가가 지정되어 있든지 파일이 존재하면 open은 실패하고, errno는 EEXIST를 갖게된다.

Determining file accessibility with access

#include <unistd.h>

Int access(const char *pathname, int amode);

- ruid에 준하여 프로세스가 파일에 접근가능한지를 알아본다.
- S_ISUID 비트가 어떻게 지정되어있는가와 상관이 없다.
- R_OK, W_OK, X_OK, F_OK(존재 여부만)
- Return 값 0이면 접근가능, -1이면 접근 불가
- 이 때 errno는
  - EACCESS: 파일 허가가 요구된 접근을 허용하지 않는다
  - ENOENT: 파일이 존재하지 않는다.

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
main()
{
  char *filename="afile";
  if(access(filename,R_OK)==-1){
          fprintf(stderr,"User cannot read file %s₩n",filename);
          exit(1);
          }
  printf("%s readable,proceeding₩n",filename);
  /* 프로그램의 나머지 부분 */
}
```

chmod **and** chown

- chmod
  - 파일 허가 변경
  - 파일의 소유자나 수퍼 유저만 할 수 있다.
- chown
  - 파일의 소유자와 그룹을 함께 변경
  - 호출 프로세스의 euid와 ruid가 일치해야 한다.
  - 불법적 시도는 EPERM 오류
  - 파일 소유자나 수퍼 유저만 할 수 있다.
  - 일반 소유자가 파일 소유권을 넘겨 줄 수 있는데 한번 변경하면 취소할 수 없다
    - 원래 사용자의 uid와 파일의 uid가 달라지기 때문에
  - 소유권이 변경된 파일의 set-user-id와 set-group-id 허가는 꺼진다.

2 Files with multiple names

- link

  link(“/usr/keith/chap.2”,”/usr/ben/2.chap”);

  - UNIX의 파일은 하나 이상의 이름으로 식별될 수 있다
  - Hard link
  - Link count

- unlink
  - 지명된 링크를 제거하고 파일의 link count를 하나 줄인다.
  - Link count가 0이 되고 현재 그 파일을 open하고 있는 프로그램이 없으면 그 파일은 시스템에서 제거 된다.

```
#include<stdlib.h>
#include<stdio.h>
#include<unistd.h>
char *usage="usage:move file1 file2 ₩n";
main(int argc,char **argv)
{
  if(argc !=3) {
            fprintf(stderr,usage);
            exit(1);
            }
  if(link(argv[1],argv[2])==-1){
            perror("link failed");
            exit(1);
            }
  if(unlink(argv[1])==-1){
            perror("unlink failed");
            exit(1);
            }
  printf("Succeeded₩n");
  exit(0);
}
```

Symbolic links

- Link의 제한
  - 사용자가 디렉토리에 link를 생성하는 것은 불가능하다.
  - 파일 시스템을 가로질러 다른 파일 시스템에 있는 파일에 대해서 link를 생성할 수 없다.
- Symbolic link
  - 그 자신이 하나의 파일
  - 자신이 link되어 있는 파일에 대한 경로를 수록, 포인터 개념
- readlink
  - Symbolic link file을 open하면 링크된 realname까지 경로를 따라가서 open 한다.
  - symname 자체의 내용을 보려면 readlink를 호출한다.

3 Obtaining file information: stat and fstat

```
#include <sys/types.h>
#include <sys/stat.h>
int stat(const char *pathname, struct stat *buf);
int fstat(int filedes, struct stat *buf);
struct stat s;
```

```
int filedes, retval;

filedes = open("/tmp/dina", O_RDWR);

retval = stat("/tmp/dina", &s);
또는
retval = fstat(filedes, &s);
```

lstat(…)

symbolic link의 경우 연결된 파일이 아니라 symbolic link 그 자체에 대한 정보를 얻는다.

Structure stat **in** <sys/stat.h>

```
struct     stat {
   dev_t  st_dev;
   long    st_pad1[3];       /* reserve for dev expansion, */
                             /* sysid definition */
   ino_t   st_ino;           /* I-node number (serial number) */
   mode_t st_mode;           /* file type and mode (permissions) */
   nlink_t st_nlink;         /* number of link */
   uid_t   st_uid;           /* user ID of owner */
   gid_t   st_gid;           /* group ID of owner */
   dev_t  st_rdev;
   long    st_pad2[2];
   off_t    st_size;         /* size in bytes, for regular files */
   long    st_pad3;          /* reserve pad for future off_t expansion */
```

```
    timestruc_t st_atime;   /* time of last access */
    timestruc_t st_mtime;  /* time of last modification */
    timestruc_t st_ctime;   /* time of last file status change */
    long    st_blksize;         /* best I/O block size */
    long    st_blocks;          /* number of blocks */
    char    st_fstype[_ST_FSTYPSZ];
    long    st_pad4[8];         /* expansion area */
};

#define  S_ISFIFO(mode)         (((mode)&0xF000) == 0x1000)
#define  S_ISCHR(mode)          (((mode)&0xF000) == 0x2000)
#define  S_ISDIR(mode)          (((mode)&0xF000) == 0x4000)
#define  S_ISBLK(mode)          (((mode)&0xF000) == 0x6000)
#define  S_ISREG(mode)          (((mode)&0xF000) == 0x8000)
#define  S_ISLNK(mode)          (((mode)&0xF000) == 0xa000)
#define  S_ISSOCK(mode)         (((mode)&0xF000) == 0xc000)
```

```
Int main(int argc, char *argv[])
{
   int                              i;
   struct stat          buf;
   char                 *ptr;

   for (i = 1; i < argc; i++) {
            printf("%s: ", argv[i]);
            if (lstat(argv[i], &buf) < 0) {
                     err_ret("lstat error");
                     continue;
            }
            if      (S_ISREG(buf.st_mode))          ptr = "regular";
            else if (S_ISDIR(buf.st_mode))          ptr = "directory";
            else if (S_ISCHR(buf.st_mode))          ptr = "character special";
            else if (S_ISBLK(buf.st_mode))          ptr = "block special";
            else if (S_ISFIFO(buf.st_mode))         ptr = "fifo";
#ifdef      S_ISLNK
            else if (S_ISLNK(buf.st_mode))          ptr = "symbolic link";
#endif
#ifdef      S_ISSOCK
            else if (S_ISSOCK(buf.st_mode))         ptr = "socket";
#endif
            else                                    ptr = "** unknown mode **";
            printf("%s\n", ptr);
   }
   exit(0);
}
```

Example: filedata

```
#include<stdio.h>
#include<sys/stat.h>

/* 허가 비트가 설정되어 있는지 결정하기 위해 */
static short octarray[9]={0040,0200,0100,
                          0040,0020,0010,
                          0004,0002,0001};
static char perms[10]="rwxrwxrwx";

int filedata(const char *pathname)
{
  struct stat statbuf;
  char descrip[10];
  int j;
  if(stat(pathname,&statbuf)==-1){
     fprintf(stderr,"Couldn't stt %s\n",pathname);
     return(-1);
  }
```

```
  for(j=0;j<9;j++)
  {
     if(statbuf.st_mode &octarray[j])
       descrip[j]=perms[j];
     else descrip[j]='-';
  }
  descrip[9]='\0';
  printf("\nFile %s :\n",pathname);
  printf("Size %1d bytes \n",statbuf.st_size);
  printf("User id %d, Group-id %d\n\n",
      statbuf.st_uid, statbuf.st_gid);
  printf("permissions:%s\n",descrip);
  return(0);
}
```

Example: lookout

```
#include<stdlib.h>
#include<stdio.h>
#include<sys/stat.h>
#define MFILE 10

void cmp(const char *,time_t);
struct stat sb;

main(int argc,char **argv)
{
  int j;
  time_t last_time[MFILE+1];
```

Example: lookout

```
if(argc<2)
   {
     fprintf(stderr,"Usage:lookout  filename\n");
     exit(1);
   }

   if(--argc>MFILE)
   {
     fprintf(stderr,"lookout: too many filenames\n");
     exit(1);
   }
```

Example: lookout

```
for(j=1;j<=argc;j++)
  {
    if(stat(argv[j],&sb)==-1)
    {
      fprintf(stderr,"lookout:couldn;t stat %s
\n",argv[j]);
      exit(1);
    }
    last_time[j]=sb.st_mtime;
  }

  for (;;)
  {
    for(j=1;j<=argc;j++)
      cmp(argv[j],last_time[j]);
    sleep(60);
  }
}
```

Example: lookout

```
void cmp(const char *name, time_t last)

{
  if(stat(name,&sb)==-1||sb.st_mtime !=last)
  {
    fprintf(stderr,"lookout:%s changed \n",name);
    exit(0);
  }
}
```

chmod **revisited**

Example: addx

```
#define XPERM 0100
main(int argc, char **argv)
{
  int k;
  struct stat statbuf;

  for(k=1;k<argc; k++)
  {
    if(stat(argv[k], &statbuf) == -1)
    {
      fprintf(stderr, "addx: couldn't stat %s\n", argv[k]);
      continue;
    }
    statbuf.st_mode |=XPERM;
    if(chmod(argv[k], statbuf.st_mode) == -1)
    fprintf(stderr, "addx: couldn't change mode for %s\n",
argv[k]);
  }
  exit(0);
}
```