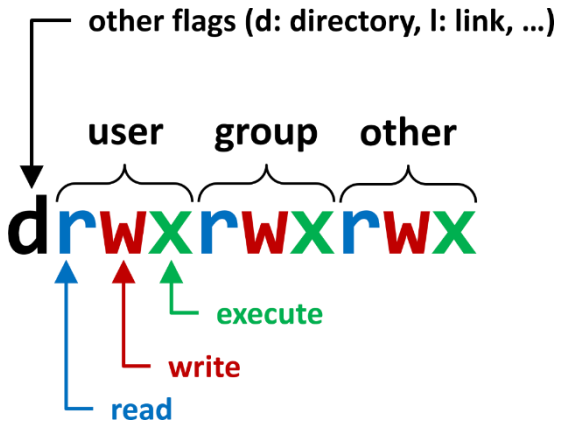
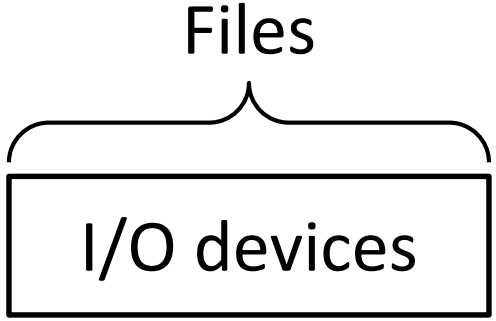
Input/Output

Unix Filesystem Concepts



Module Outline

- The Unix File Concept
- The Unix Filesystem
- Filesystems and Security
- Module Summary



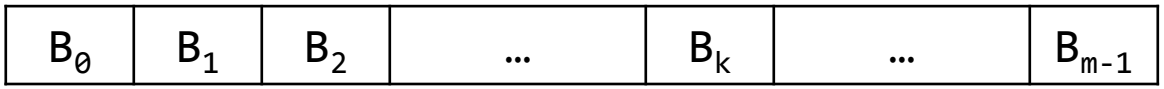
I/O device abstraction

The Unix File Concept

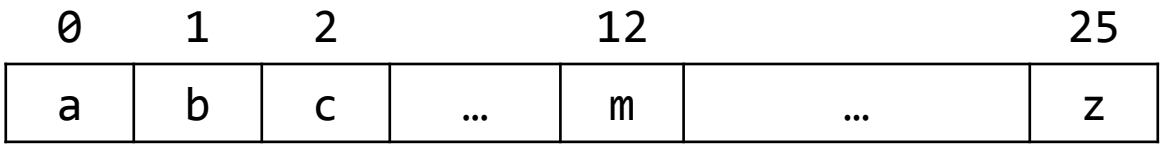
Unix Files

- **A Unix file is a sequence of m bytes:**
  - B_0 , B_1 , B_2 , ... , B_k , ... , B_{m-1}

file system translate this bunch of bytes to meaning for information format type



- Example: File containing the lower-case English alphabet a-z



Unix Files

- in *nix systems, ***everything*** is modeled as a file
  - file system
  - disk, disk partitions
  - memory
  - USB devices: keyboard, mouse, …
  - display
  - network
  - audio
  - video
  - …

ls

Unix Files: Examples

file 끼리 묶기. (file system)

- I/O devices are represented as files: /dev
  - /dev/sda (first disk on SATA bus)
  - /dev/input/mice (aggregate of all connected mice)
  - /dev/tty2 (terminal)

) cpu, input...

- The kernel is exposed with a number of files
  - /dev/kmem (kernel memory image)
  - /proc (process) (kernel data structures) : virtual file system (kernel data struc)

    not a real file...
- System configuration is mapped as files
  - /sys

cat mem | hexdump -C (| less)

maps (memory addresses (virtual))

Unix File Types

- **Regular file**
  - File containing user/app data (binary, text, whatever)
  - OS does not know anything about the format
    - other than “sequence of bytes”, akin to main memory
- **Directory file**
  - A file that contains the names and locations of other files
- **Character special and block special files**
  - Terminals (character special) and disks (block special)

unit of byte (char)

unit of block not bytes (char)
- **FIFO (named pipe)**
  - A file type used for inter-process communication
- **Socket**
  - A file type used for (local or networked) communication between processes

Unix I/O

- **Key Features**
  - Design concept: All input and output is handled in a consistent and uniform way
  - Elegant mapping of files to devices allows kernel to export simple interface called Unix I/O

One single file interface to interact with any kind of device

(well, almost)

The Unix Filesystem

The Unix Filesystem

- **One root to rule them all**
  - single file system starting with at the root (“/”)
  - unlike Windows, there is no concept of a “drive”
  - additional filesystems are mapped into the file system tree as a directory
  - mount point = directory where a filesystem is attached

```
devel@csapvm $ ls -1
share
temp
work
devel@csapvm $ mkdir extern
devel@csapvm $ echo "hello" > extern/hello
devel@csapvm $ ls extern/
hello
devel@csapvm $ sudo mount –t tmpfs /dev/shm extern/
Password:
devel@csapvm $ ls extern/
devel@csapvm $ echo "extern" > extern/extern
devel@csapvm $ ls extern/
extern
devel@csapvm $ sudo umount extern
devel@csapvm $ ls extern/
hello
devel@csapvm
```

The Unix Filesystem

What is mount

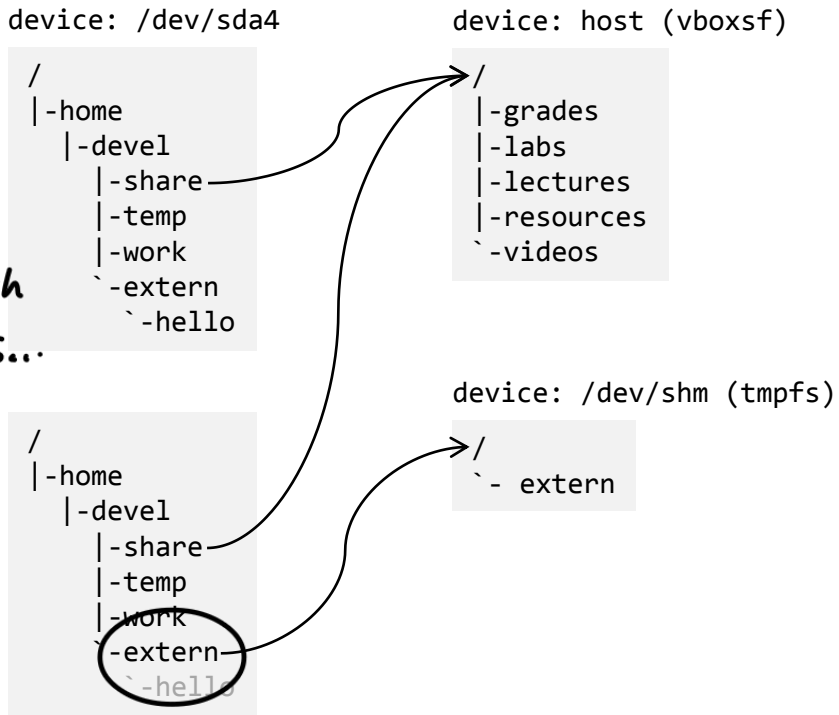
- **One root to rule them all**
  - mapped filesystems hide the contents of the directory tree under the mount point

```
devel@csapvm $ ls -1
share
temp
work
devel@csapvm $ mkdir extern
devel@csapvm $ echo "hello" > extern/hello
devel@csapvm $ ls extern/
hello
devel@csapvm $ sudo mount -t tmpfs /dev/shm extern/
Password:
devel@csapvm $ ls extern/
devel@csapvm $ echo "extern" > extern/extern
devel@csapvm $ ls extern/
extern
devel@csapvm $ sudo umount extern
devel@csapvm $ ls extern/
hello
devel@csapvm
```

repeat

create dir

Link holder with others...



The Unix Filesystem

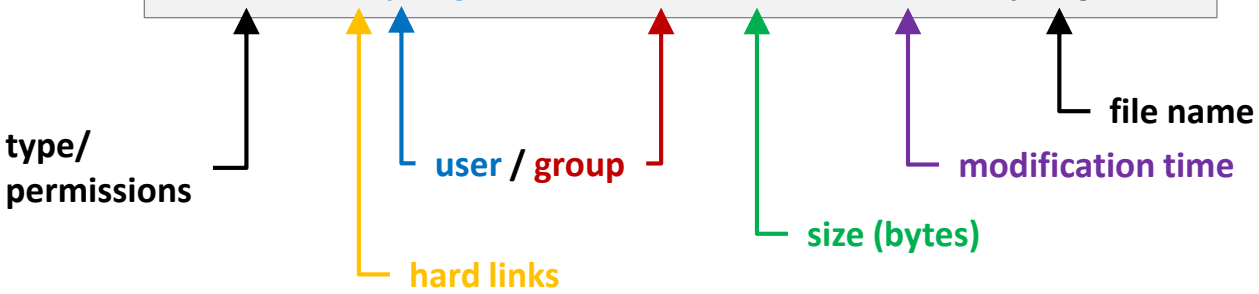
- **One root to rule them all**
  - extremely powerful concept
  - each mounted filesystem can have additional properties (Auth)
    - do not allow writes (ro)
    - do not update access time (noatime)
    - do not allow execution of programs (noexec)
    - do not allow set user/group id (nosuid)
    - …

```
devel@csapvm $ mount
/dev/sda4 on / type ext4 (rw,noatime)
...
none on /tmp type tmpfs (rw,noatime,size=262144k)
none on /var/tmp type tmpfs (rw,noatime,size=131072k)
devel_share on /home/devel/share type vboxsf (rw,nodev,relatime,iocharset=utf8,uid=1000,gid=100)
...
```

The Unix Filesystem

- **User, group, soft & hard links**
  - each file is owned by a *user* and a *group*

```
bernhard@intel72:/home$ ls -l
total 148
drwxr-xr-x 13 root      root     4096 Feb  4  2017 alumni
drwxr-xr-x  4 anna      student  4096 Aug 31  2016 anna
-rw-------  1 root      root     8192 Jan 20  2017 aquota.user
drwxr-xr-x  8 barend    student  4096 Dec 19  2018 barend
drwxr-xr-x 16 bernhard  prof     4096 Sep 16 00:47 bernhard
drwxr-xr-x  7 camilo    student  4096 Jun 28  2017 camilo
drwxr-xr-x  6 changmin  student  4096 Mar 23  2017 changmin
drwxr-xr-x 13 changyeon postdoc  4096 Feb 19  2021 changyeon
...
drwxr-xr-x  4 simeon    student  4096 Mar 10  2017 simeon
drwxr-xr-x  5 root      root     4096 Apr 21  2020 svn
drwxr-xr-x  6 vm        users    4096 May 10 10:43 vm
drwxr-xr-x  5 vmail     vmail    4096 Apr 19  2012 vmail
drwxr-xr-x  4 xinyi     student  4096 Aug 31  2016 xinyi
drwxr-xr-x 20 younghyun postdoc  4096 Jun 18 13:56 younghyun
drwxr-xr-x  8 youngsu   student  4096 Jun 17  2019 youngsu
```



The Unix Filesystem

- **User, group, soft & hard links**
  - files can point to each other via soft or hard links

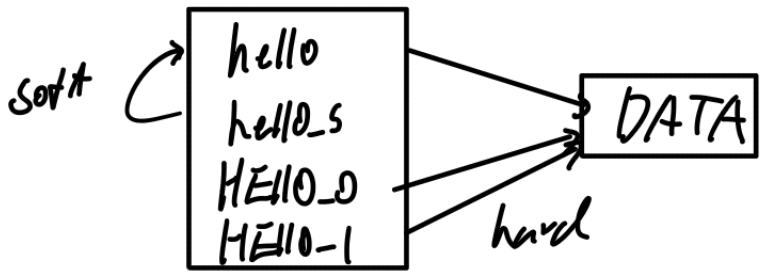
```
$ printf "hello" > hello.txt
$ printf "test" > test.txt
$ ls -l
total 8
-rw-r--r-- 1 devel users 5 Sep 27 01:10 hello.txt
-rw-r--r-- 1 devel users 4 Sep 27 01:11 test.txt

$ ln hello.txt helloworld.txt
$ ln -s test.txt testing
$ ls -l
total 12
-rw-r--r-- 2 devel users 5 Sep 27 01:10 hello.txt
-rw-r--r-- 2 devel users 5 Sep 27 01:10 helloworld.txt
lrwxrwxrwx 1 devel users 8 Sep 27 01:12 testing -> test.txt
-rw-r--r-- 1 devel users 4 Sep 27 01:11 test.txt
```

ln: create hard link

ln -s: create soft (symbolic) link

hard link count

l: soft (symbolic) link flag



The Unix Filesystem

- **User, group, soft & hard links**
  - files can point to each other via soft or hard links

Analysis ls -la
Any At home.

```
$ printf "hello" > hello.txt
$ printf "test" > test.txt
$ ls -l
total 8
-rw-r--r-- 1 devel users 5 Sep 27 01:10 hello.txt
-rw-r--r-- 1 devel users 4 Sep 27 01:11 test.txt

$ ln hello.txt helloworld.txt
$ ln -s test.txt testing
$ ls -l
total 12
-rw-r--r-- 2 devel users 5 Sep 27 01
-rw-r--r-- 2 devel users 5 Sep 27 01
lrwxrwxrwx 1 devel users 8 Sep 27 01
-rw-r--r-- 1 devel users 4 Sep 27 01
```

```
…

$ printf ", world!"\\n >> helloworld.txt

$ cat helloworld.txt
hello, world!
$ cat hello.txt
hello, world!

$ ls -l
total 12
-rw-r--r-- 2 devel users 14 Sep 27 01:12 hello.txt
-rw-r--r-- 2 devel users 14 Sep 27 01:12 helloworld.txt
lrwxrwxrwx 1 devel users  8 Sep 27 01:12 testing -> test.txt
-rw-r--r-- 1 devel users  4 Sep 27 01:11 test.txt
```

copy.

The Unix Filesystem

- **User, group, soft & hard links**
  - files can point to each other via soft or hard links

```
$ printf "hello" > hello.txt
$ printf "test" > test.txt
$ ls -l
total 8
-rw-r--r-- 1 devel users 5 Sep 27 01:10 hello.txt
-rw-r--r-- 1 devel users 4 Sep 27 01:11 test.txt

$ ln hello.txt helloworld.txt
$ ln -s test.txt testing
$ ls -l
total 12
-rw-r--r-- 2 devel users 5 Sep 27 01
-rw-r--r-- 2 devel users 5 Sep 27 01
lrwxrwxrwx 1 devel users 8 Sep 27 01
-rw-r--r-- 1 devel users 4 Sep 27 01
```

```
…

$ printf ", test, and test!"\\n >> testing

$ cat test.txt
test, test, and test!
$ cat testing
test, test, and test!

$ ls -l
total 12
-rw-r--r-- 2 devel users 14 Sep 27 01:12 hello.txt
-rw-r--r-- 2 devel users 14 Sep 27 01:12 helloworld.txt
lrwxrwxrwx 1 devel users  8 Sep 27 01:12 testing -> test.txt
-rw-r--r-- 1 devel users 22 Sep 27 01:13 test.txt
```

The Unix Filesystem

- **User, group, soft & hard links**
  - files can point to each other via soft or hard links

```
$ printf "hello" > hello.txt
$ printf "test" > test.txt
$ ls -l
total 8
-rw-r--r-- 1 devel users 5 Sep 27 01:10 hello.txt
-rw-r--r-- 1 devel users 4 Sep 27 01:11 test.txt

$ ln hello.txt helloworld.txt
$ ln -s test.txt testing
$ ls -l
total 12
-rw-r--r-- 2 devel users 5 Sep 27 01
-rw-r--r-- 2 devel users 5 Sep 27 01
lrwxrwxrwx 1 devel users 8 Sep 27 01
-rw-r--r-- 1 devel users 4 Sep 27 01
```

```
…

$ rm hello.txt
$ rm test.txt

$ ls -l
total 4
-rw-r--r-- 1 devel users 14 Sep 27 01:12 helloworld.txt
lrwxrwxrwx 1 devel users  8 Sep 27 01:12 testing -> test.txt

$ cat helloworld.txt
hello, world!

$ cat testing
cat: testing: No such file or directory
```

broken (soft) link

The Unix Filesystem

- **File types**
  - Unix knows different types of files

```
/tmp/demo $ ls -l
total 8
drwxr-xr-x  2  bernhard       users    40  Sep  3 20:00  directory
prw-r--r--  1  bernhard       users     0  Sep  3 20:01  fifo
-rw-r--r--  2  bernhard       users    14  Sep  3 20:03  file
-rw-r--r--  2  bernhard       users    14  Sep  3 20:03  hardlink
lrwxrwxrwx  1  bernhard       users     4  Sep  3 20:00  softlink -> file
```

Letter	File type
-	regular file
d	directory
l	soft link (symbolic link)
p	named pipe (fifo)
s	socket
c	character device file (see /dev)
b	block device file (see /dev) only I/O with unit of block.

The Unix Filesystem

- **Hidden files**
  - Hidden files start with a "." in Unix file systems

```
/tmp/demo $ ls -la
total 8
drwxr-xr-x  4  bernhard      users   180  Sep  3 20:03  .
drwxrwxrwt 18  root          root   1120  Sep  3 19:59  ..
drwxr-xr-x  2  bernhard      users    40  Sep  3 20:00  directory
prw-r--r--  1  bernhard      users     0  Sep  3 20:01  fifo
-rw-r--r--  2  bernhard      users    14  Sep  3 20:03  file
-rw-r--r--  2  bernhard      users    14  Sep  3 20:03  hardlink
drwxr-xr-x  2  bernhard      users    40  Sep  3 20:00 .hiddendir
-rw-r--r--  1  bernhard      users     0  Sep  3 20:00 .hiddenfile
lrwxrwxrwx  1  bernhard      users     4  Sep  3 20:00  softlink -> file
```

  - Special (hidden) entries
    - "." current directory
    - ".." parent directory (hence "cd ..")

Filesystem Hierarchy Standard (FHS)

- **What goes where?**
  - Is it all a big mess?

```
$ $ ls -l /
total 76
drwxr-xr-x   2 root root  4096 Sep  3 02:11 bin
drwxr-xr-x   3 root root  4096 Aug  1 20:37 boot
drwxr-xr-x  20 root root  4020 Sep  1 23:17 dev
drwxr-xr-x  88 root root  4096 Sep  3 11:59 etc
drwxr-xr-x   5 root root  4096 Mar 24  2022 home
drwxr-xr-x  13 root root  4096 Jul  5 21:40 lib
drwxr-xr-x   7 root root  4096 Sep  3 02:11 lib64
drwx------   2 root root 16384 Mar 23  2022 lost+found
drwxr-xr-x   2 root root  4096 Mar 21  2022 media
drwxr-xr-x   4 root root  4096 Sep 16  2022 mnt
drwxr-xr-x  16 root root  4096 Jul 16 19:05 opt
dr-xr-xr-x 447 root root     0 Aug 19 21:52 proc
drwx------   7 root root  4096 Sep  3 11:59 root
drwxr-xr-x  19 root root   740 Sep  1 23:16 run
drwxr-xr-x   2 root root 12288 Sep  3 02:11 sbin
dr-xr-xr-x  12 root root     0 Aug 19 21:52 sys
drwxrwxrwt  17 root root  1100 Sep  3 18:51 tmp
drwxr-xr-x  12 root root  4096 Sep 20  2022 usr
drwxr-xr-x  10 root root  4096 Sep  3 01:37 var
```

Filesystem Hierarchy Standard (FHS)

- **The Filesystem Hierarchy Standard (FHS)**
  - Conventions for the layout of directories and files on Unix systems
  - Maintained by the Linux Foundation: https://refspecs.linuxfoundation.org/fhs.shtml
    - initial version released February 14, 1994, current version: 3.0 (June 3, 2015)

Directory	Description
/	**root**
/bin	Essential binaries required during boot-up
/boot	boot loader, kernel
/dev	Device files (disks, partitions, memory, audio, video, …)
/etc	Host-specific, system-wide configuration files
/home	User home directories
/lib[64]	System libraries (required by binaries in /bin, /sbin)
/media, /mnt	Mount points for removable media
…	

Filesystem Hierarchy Standard (FHS)

- **The Filesystem Hierarchy Standard (FHS)**
  - https://refspecs.linuxfoundation.org/fhs.shtml

Directory	Description
…	
/opt	Additional application software packages
/proc	Process and kernel information (virtual filesystem)
/root	Home directory for the root user (administrator)
/run	Run-time variable data
/sbin	Essential system binaries
/sys	Device driver & kernel information and configuration
/tmp, /var/tmp	Temporary files (often not preserved across reboots)
/usr	Secondary hierarchy for read-only user data
/var	Variable files

Filesystem Hierarchy Standard (FHS)

- **The Filesystem Hierarchy Standard (FHS)**
  - https://refspecs.linuxfoundation.org/fhs.shtml

Directory	Description
/usr	**Secondary hierarchy for read-only user data**
/usr/bin, /usr/sbin	Non-essential binaries
/usr/include	Standard include files (C headers)
/usr/lib[64]	Libraries required by binaries in /usr/bin, /usr/sbin
/usr/libexec	Binaries run via scripts (do not run directly)
/usr/local	Tertiary hierarchy for local data specific to this machine
/usr/share	Applicate shared data
/usr/src	Source code (kernel source)

Filesystem Hierarchy Standard (FHS)

- **The Filesystem Hierarchy Standard (FHS)**
  - https://refspecs.linuxfoundation.org/fhs.shtml

Directory	Description
/var	**Variable files**
/var/cache	Application cache data
/var/db	Gentoo portage (config & source files)
/var/lib	Persistent state data modified by applications
/var/lock	Lock files to keep track of resources currently in use
/var/log	System log files
/var/mail	Mailbox files (servers only)
/var/run	Run-time variable data (FHS 3.0: mapped to /run)
/var/spool	Spool for tasks waiting to be processed
/var/tmp	Temporary files (often not preserved across reboots)

Filesystem Hierarchy Standard (FHS)

- **The Filesystem Hierarchy Standard (FHS)**
  - https://refspecs.linuxfoundation.org/fhs.shtml

Directory	Description
/home/<USER>/	**User-specific files**
.cache	User cached data
.config	User-specific configuration
.local	User local data (bin, lib, share)
.mozilla	Application-specific configuration, z.B. for Mozilla
.ssh	idem for SSH
.vim	and for vim
	plus user-created directories and files

  - Note: "~" is an abbreviation for /home/<USER>
    - ls ~/.config

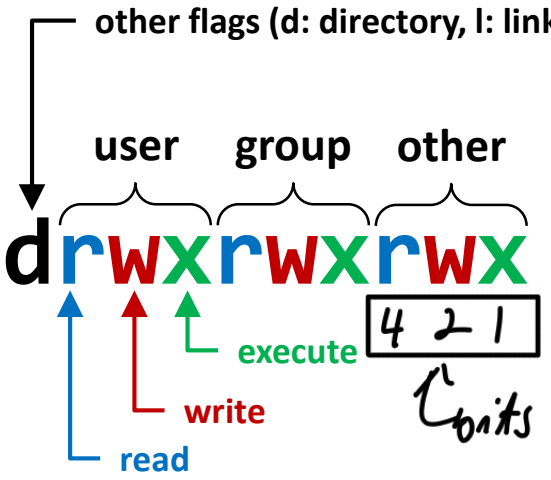
```
fstatat(dd, entry->d_name, &sb, AT_SYMLINK_NOFOLLOW);

if (S_ISREG(sb.st_mode) &&
    (((sb.st_uid == 0) && (sb.st_mode & S_ISUID)) ||
     ((sb.st_gid == 0) && (sb.st_mode & S_ISGID)))
{
  fstatfs(dd, &dsb);
  if (!(dsb.f_flags & (ST_NOEXEC|ST_NOSUID))) {
    // dangerous configuration
  }
}
```

Filesystems and Security

Filesystems and Security

- **Standard *nix Access Control Lists (ACL)**
  - three levels of access
    - owner, group, other
  - three kinds of permissions
    - read (r), write (w), execute (x)

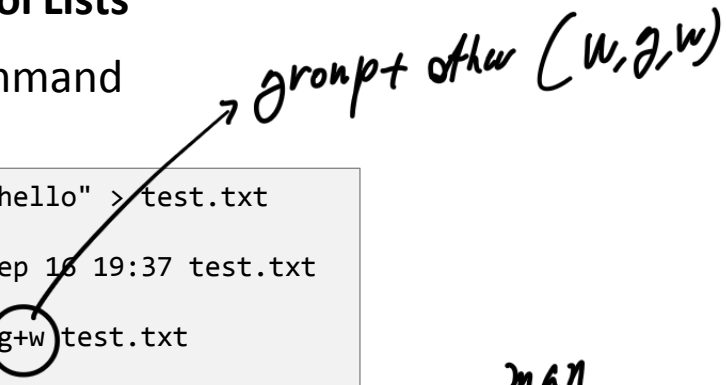
```
devel@csapvm $ ls -l
total 60
drwxr-xr-x 2 devel devel  4096 Sep 26 19:44 doc
-rwxr-xr-- 1 devel devel 18440 Sep 27 01:50 lsec
-rw-r--r-- 1 devel devel  9842 Sep 27 00:52 lsec.c
-rw-r--r-- 1 devel devel  5493 Sep 27 00:37 lsec.h
-rw-rw-rw- 1 devel devel  1009 Sep 26 19:44 Makefile
-r--r----- 1 devel devel 22738 Sep 26 19:44 README.md
drwxr-xr-x 2 devel devel  4096 Sep 26 19:44 reference
drwx---r-x 2 devel devel  4096 Sep 26 19:44 tools
```



e**X**ecute permission on
- file: execute (run) program
- directory: list contents of directory

Filesystems and Security

- **Standard *nix Access Control Lists**
  - modify with chmod command




```
devel@csapvm ~/temp $ echo "hello" > test.txt
devel@csapvm ~/temp $ ls -l
-rw-r--r-- 1 devel devel 6 Sep 16 19:37 test.txt

devel@csapvm ~/temp $ chmod g+w test.txt
devel@csapvm ~/temp $ ls -l
-rw-rw-r-- 1 devel devel 6 Sep 16 19:37 test.txt

devel@csapvm ~/temp $ echo "ls -l" > script.sh
devel@csapvm ~/temp $ ./script.sh
-bash: ./script.sh: Permission denied
devel@csapvm ~/temp $ ls -l
-rw-r--r-- 1 devel devel 6 Sep 16 19:38 script.sh

devel@csapvm ~/temp $ chmod 750 script.sh
devel@csapvm ~/temp $ ls -l
-rwxr-x--- 1 devel devel 6 Sep 16 19:37 test.txt
```

man
manual

  - for details, see man 1 chmod

Filesystems and Security

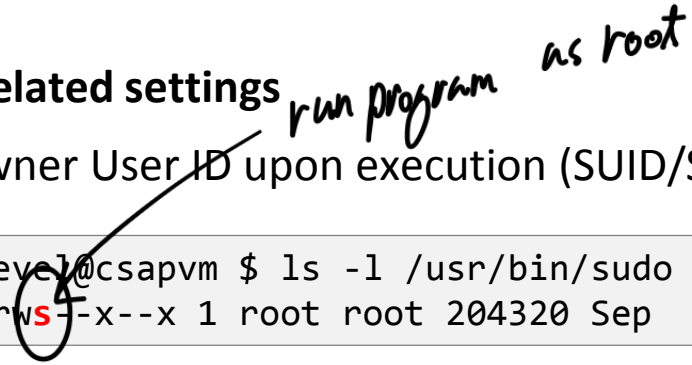
- **Security-related settings**
  - Sticky bit (Not super important)

```
devel@csapvm $ ls -ld /tmp
drwxrwxrwt 6 root root 240 Oct  8 14:32 /tmp
```

    - files in a directory with the sticky bit set can be renamed or deleted only by the owner of the file, by the owner of the directory, and by a privileged user/process
      - should be set on world-writable directories such as /tmp
    - flag available in struct stat (S_ISVTX)
      - man 2 stat
      - man 7 inode

Filesystems and Security

- **Security-related settings**
  - Set owner User ID upon execution (SUID/SGID bit)




```
devel@csapvm $ ls -l /usr/bin/sudo
-rwsr-x--x 1 root root 204320 Sep  1 16:17 /usr/bin/sudo
```

    - binary executed with permissions of owner (root in this case)
      (as opposed to context of user who executes it)
      - useful to give temporary permissions of owner
      - only trusted binaries owned by root must have suid/sgid bits set!
    - flag available in struct stat (`S_ISUID` / `S_ISGID`): man 2 stat, man 7 inode
    - filesystem may disallow SUID:
      use `fstatfs()` and check for `ST_NOSUID` flag

Filesystems and Security

skipped

- **Security-related settings**
  - World-writable directory with execute permission

```
devel@csapvm $ ls -ld /tmp
drwxrwxrwx 6 root root 240 Oct  8 14:32 /tmp
or
drwxrwxrwt 6 root root 240 Oct  8 14:32 /tmp
```

    - world-writable directories on a file system with execute permission are a security risk
    - anyone with access to the system may place an executable and run it (typical scenario: webserver breach → write script to /tmp → execute it)
    - world-writable flag in `struct stat` (S_IWOTH)
    - filesystem may disallow execution: use `fstatfs()` and check for ST_NOEXEC flag

Example: Explore SUID/GUID

- **Set user/group owner**

  `sudo chown <usr>[:<grp>]`

- **Set suid/sgid bit**

  suid bit

  `sudo chmod 4755 <exe>`

  sgid bit

  `sudo chmod 2755 <exe>`

  suid+sgid bits

  `sudo chmod 6755 <exe>`

```
devel@csapvm ~/work/03 $ gcc -o wai whoami.c
devel@csapvm ~/work/03 $ ./wai
User & group information
------------------------
  User:             devel           (1000)
  Group:            devel           (1000)

  Effective user:   devel           (1000)
  Effective group:  devel           (1000)

devel@csapvm ~/work/03 $ sudo chown tester:users wai
devel@csapvm ~/work/03 $ sudo chmod 4755 wai
devel@csapvm ~/work/03 $ ls -l
total 20
-rwsr-xr-x 1 tester users 16000 Sep 27 02:32 wai
-rw-r--r-- 1 devel  devel  1380 Sep 27 02:31 whoami.c

devel@csapvm ~/work/03 $ ./wai
User & group information
------------------------
  User:             devel           (1000)
  Group:            devel           (1000)

  Effective user:   tester          (1001)
  Effective group:  devel           (1000)
```

Example: Explore SUID/GUID

- **whoami.c: print user/group**
  - **effective user/group**
    user/group under whose permission the process is executed
  - **(real) user/group**
    original user/group under which the process was started (before suid/sgid)

```c
#include <grp.h>
#include <pwd.h>
…
int main(int argc, char *argv[]) {
  // get user id, effective user id, group id, effective group id
  uid_t uid = getuid();
  uid_t euid = geteuid();
  gid_t gid = getgid();
  gid_t egid = getegid();

  // get user and group names
  char *user, *euser, *group, *egroup;
  struct passwd *pwd;
  struct group *grp;

  if ((pwd = getpwuid(uid)) != NULL) user = strdup(pwd->pw_name);
  if ((pwd = getpwuid(euid)) != NULL) euser = strdup(pwd->pw_name);
  if ((grp = getgrgid(gid)) != NULL) group = strdup(grp->gr_name);
  if ((grp = getgrgid(egid)) != NULL) egroup = strdup(grp->gr_name);

  // print results
  printf("User & group information\n"
         "------------------------\n"
         "  User:             %-16s (%4d)\n"
         "  Group:            %-16s (%4d)\n"
         "\n"
         "  Effective user:   %-16s (%4d)\n"
         "  Effective group:  %-16s (%4d)\n",
         user ? user : "n/a", uid,
         group ? group : "n/a", gid,
         euser ? euser : "n/a", euid,
         egroup ? egroup : "n/a", egid);

  // free allocated memory
  free(user); free(egroup); free(group); free(euser);

  return EXIT_SUCCESS;
}
```

whoami.c

Example: Checking for SUID/SGID Bit

```
DIR *d = opendir(name);
int dd = dirfd(d);
struct dirent *entry;

while ((entry = getNext(d)) != NULL) {
  struct stat sb;
  struct statfs dsb;

  fstatat(dd, entry->d_name, &sb,                    // get metadata of directory entry
          AT_SYMLINK_NOFOLLOW);

  if (S_ISREG(sb.st_mode) &&                         // if it's a regular file and
      (((sb.st_uid == 0) && (sb.st_mode & S_ISUID)) || // the user is root & SUID is set or
       ((sb.st_gid == 0) && (sb.st_mode & S_ISGID)))   // the group is root & SGID is set
  {
    fstatfs(dd, &dsb);                               // get metadata of file system
    if (!(dsb.f_flags & (ST_NOEXEC|ST_NOSUID))) {    // if neither NOEXEC nor NOSUID are set
      // dangerous configuration                     //   then this is potentially dangerous
    }
  }

  …
}
```

- for readability, no error checking performed.

Extended File System Security Concepts

- **POSIX ACLs are limited to access permissions for the user, a group, and everybody else**

- **Extended file attributes (xattrs) provide an extensible and more flexible way to store meta data (including ACLs) about a file**
  - xattrs are key=value pairs where
    - key has the form “namespace.attribute”
    - and value is a string
  - currently, xattrs defines the namepaces “security”, “system”, “trusted”, and “user”
    - security: used by kernel modules such as SELinux to implement advanced ACLs
    - system: used by the kernel to store system objects
    - trusted: attributed only visible to processes with the CAP_SYS_ADMIN capability
    - user: store arbitrary additional information about a file such as its mime type, md5sum, character encoding, etc.

Extended File System Security Concepts

- Check whether a filesystem supports xattrs

```
devel@csapvm $ mount | grep "/ "
/dev/sda4 on / type ext4 (rw,noatime)
devel@csapvm $ cat /proc/fs/ext4/sda4/options  | grep xattr
user_xattr
```

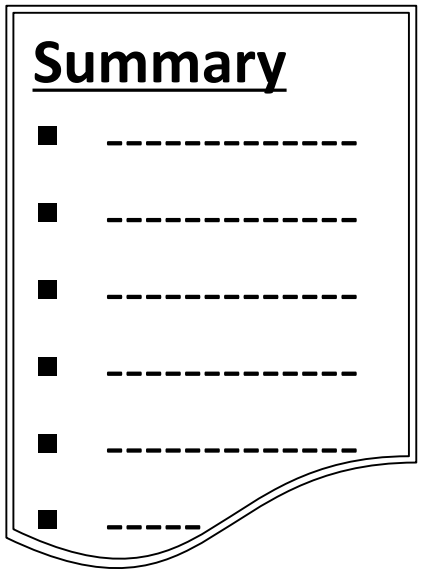
- Set/get ACLs with setfacl / getfacl

```
devel@csapvm $ echo "Hello" > file.txt
devel@csapvm $ ls -l file.txt
-rw-r--r-- 1 devel devel 6 Mar  7 19:75 file.txt
devel@csapvm $ setfacl -m u:svn:r file.txt
devel@csapvm $ ls -l file.txt
-rw-r--r--+ 1 devel devel 6 Mar  7 19:75 file.txt
devel@csapvm $ getfacl file.txt
# file: file.txt
# owner: devel
# group: devel
user::rw-
user:svn:r--
group::r--
mask::r--
other::r--
```

Extended File System Security Concepts

- Set/get arbitrary attributes with setfattr/getfattr

```
devel@csapvm $ md5sum file.txt
09f7e02f1290be211da707a266f153b3  file.txt
devel@csapvm $ setfattr -n user.checksum.md5 -v
09f7e02f1290be211da707a266f153b3 file.txt
devel@csapvm $ ls -l
total 8
-rw-r--r--+ 1 devel devel 6 Mar  7 19:75 file.txt
devel@csapvm $ getfattr file.txt
# file: file.txt
user.checksum.md5

devel@csapvm $ getfattr -n user.checksum.md5 file.txt
# file: file.txt
user.checksum.md5="09f7e02f1290be211da707a266f153b3“
devel@csapvm $ setfattr -x user.checksum.md5 file.txt
devel@csapvm $ getfattr file.txt
devel@csapvm $ getfattr -n user.checksum.md5 file.txt
file.txt: user.checksum.md5: No such attribute
```



Module Summary

Summary

- Unix concept: everything is a file
- Filesystems support many “advanced” features
  - mount points of different filesystems under common root
  - mounting with different permissions
  - hard and soft links
  - set user/group id
- Security
  - Access Control Lists (ACL)
    - for user, group, and other (=everybody else)
    - read, write, and execute permission
  - many “dangerous” configurations possible, especially sticky, suid/sgid bits
  - Extended file attributes (xattrs) provide finer-grained control