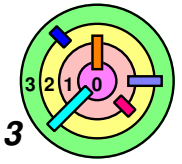
# Access Protection

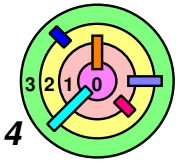
- OS needs to make sure that only *authorized processes* are allowed access to system resources
  - various ways to provide this
- Unix (and many other systems, such as Windows) associates with files some indication of which *security principals* (i.e., the "who") are allowed access
  - along with what sort of access is allowed (i.e., the "what")
- A *security principal* is normally *a user* or *a group of users*
  - a file typically contains two pieces of information
    - which *user* owns the file (`uid`)
    - which *group* owns the file (`gid`)
  - each running process can have several security principals associated with it
    - for Sixth-Edition Unix, one user ID and one group ID
    - a process in some other OSes can have more than one group IDs



# Access Protection

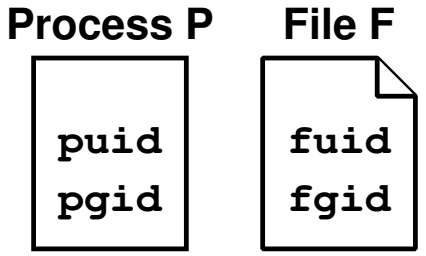
```
% ls -l z
-rw-r--r--  1 bill   adm      593 Dec 17 13:34 z
```

- Each file has associated with it a set of access permissions
  - there are 3 classes of security principals:
    - *user:* owner of the file
    - *group:* group owner of the file
    - *others:* everyone else
  - for each of the 3 classes of principals, specify what sorts of operations on the file are allowed
  - the operations are grouped into 3 classes:
    - *read:* can read a file or directory
    - *write:* can write a file or directory
    - *execute:* one must have *execute permission* for a *directory* in order to *follow a path through it*



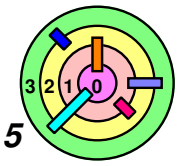
# Access Protection

- ***Rules for checking permissions***
  1) determines the ***smallest class of principals the requester belongs to*** ("user" being smallest and "others" being largest)
  2) then it checks for appropriate permissions within that class




- Can process P access file F?
  - in (1), we need to determine ***which class*** is P
    - `puid`: process `uid`
    - `pgid`: process `gid`
    - `fuid`: file `uid`
    - `fgid`: file `gid`

```
if (puid == fuid) {
    /* requester is "user/owner" */
} else if (pgid == fgid) {
    /* requester is"group-owner" */
} else {
    /* requester is"others"*/
}
```



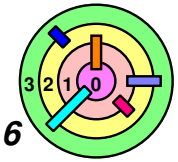
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  1) Q: May `andy` list the contents of directory `A`?



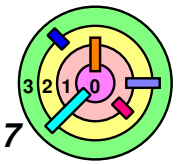
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  1) Q: May `andy` list the contents of directory `A`?
     A: No



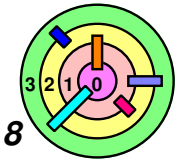
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  2) Q: May `andy` read `A/x`?



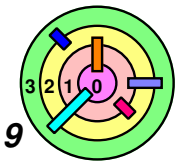
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x   2 bill    adm     1024 Dec 17 13:34 A
drwxr-----   2 bill    adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-   1 bill    adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-   1 bill    adm      446 Dec 17 13:34 x
-rw----rw-   1 trina   adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  2) Q: May `andy` read `A/x`?
     A: Yes



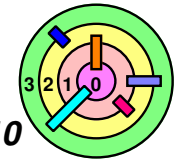
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  3) Q: May `trina` list the contents of directory `B`?



# Permissions Example

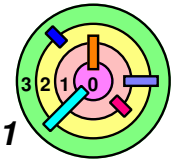
```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not

3) Q: May `trina` list the contents of directory `B`?
   A: Yes



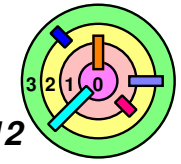
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  4) Q: May `trina` modify `B/y`?



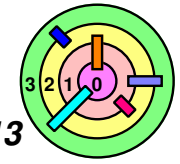
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  4) Q: May `trina` modify `B/y`?
     A: No



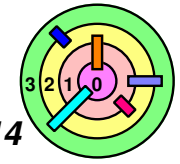
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  5) Q: May `bill` modify `B/x`?



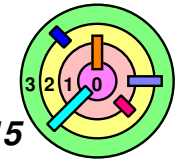
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  5) Q: May `bill` modify `B/x`?
     A: No



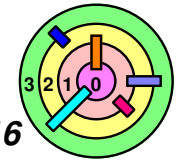
# Permissions Example

```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

- Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not
  6) Q: May `bill` read `B/y`?



# Permissions Example

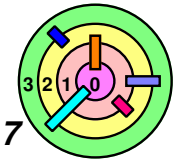
```
% ls -lR
.:
total 2
drwxr-x--x  2 bill   adm     1024 Dec 17 13:34 A
drwxr-----  2 bill   adm     1024 Dec 17 13:34 B

./A:
total 1
-rw-rw-rw-  1 bill   adm      593 Dec 17 13:34 x

./B:
total 2
-r--rw-rw-  1 bill   adm      446 Dec 17 13:34 x
-rw----rw-  1 trina  adm      446 Dec 17 13:45 y
```

Suppose that `bill` and `trina` are members of the `adm` group and `andy` is not

6) Q: May `bill` read `B/y`?
   A: No

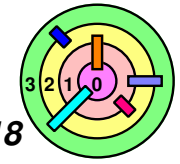


# Open

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *path, int options [, mode_t mode])
```

`options`

- O_RDONLY open for reading only
- O_WRONLY  open for writing only
- O_RDWR open for reading and writing
- O_APPEND set the file offset to end of file prior to each write
- O_CREAT if the file does not exist, then create it, setting its mode to `mode` adjusted by user mask (umask)
- O_EXCL: if O_EXCL and O_CREAT are set, then open fails if the file exists
- O_TRUNC delete any previous contents of the file
- O_NONBLOCK don't wait if I/O cannot be done immediately
- some options are not compatible with other options

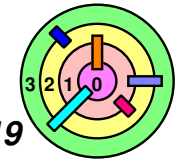


# Setting File Permissions

```
#include <sys/types.h>
#include <sys/stat.h>
int chmod(const char *path, mode_t mode)
```

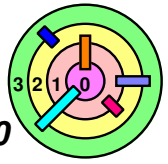
- sets the file permissions of the given file to those specified in `mode`
- only the owner of a file and the superuser may change its permissions
- nine combinable possibilities for mode (read/write/execute for user, group, and others)
- `S_IRUSR` (0400), `S_IWUSR` (0200), `S_IXUSR` (0100)
- `S_IRGRP` (040), `S_IWGRP` (020), `S_IXGRP` (010)
- `S_IROTH` (04), `S_IWOTH` (02), `S_IXOTH` (01)
  - note: numeric prefix of 0 means the number is in *octal* format

```
% chmod 0640 z
% ls -l z
-rw-r-----  1 bill   adm      593 Dec 17 13:34 z
```



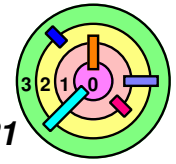
# Creating a File

- Use either `open` or `creat`
  - `open(const char *pathname, int flags, mode_t mode)`
    - flags must include O_CREAT to create a file
  - `creat(const char *pathname, mode_t mode)`
  - `open` is preferred
- The `mode` parameter helps specify the permissions of the newly created file
  - permissions = `mode & ~umask`



# Umask

- Standard programs create files with "maximum needed permissions" as *mode*
  - compilers: `0777`
  - editors: `0666`
- Per-process parameter, *umask*, used to *turn off* undesired permission bits
  - e.g., turn off all permissions for others, write permission for group: set umask to `027`
  - compilers: permissions = `0777 & ~(027) = 0750`
  - editors: permissions = `0666 & ~(027) = 0640`
  - set with `umask()` system call or (usually) `umask` shell command



# Midterm Exam Coverage

- Midterm exam covers everything from the beginning of the semester to this slide
  - Ch 1 through Ch 4 only
    - Ch 5 materials are excluded from the midterm
  - final exam coverage will not overlap midterm coverage
    - since the topics covered by the final exam is not independent of the midterm coverage, we say the final exam "focuses" on Ch 5 plus everything beyond this slide