浙江大學
ZHEJIANG UNIVERSITY

Set-UID

Yajin Zhou (http://yajin.org)

Zhejiang University

Credits: SEEDLab
http://www.cis.syr.edu/~wedu/seed/

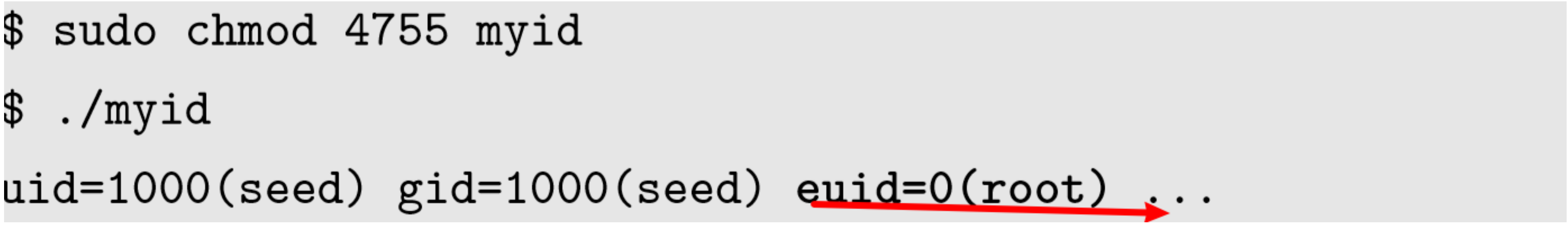
SET-UID

- A process has three IDs
  - Real user ID, effective user ID, saved user ID
  - Real user ID: the real owner of the process
  - Effective user ID: the ID used for access control

```
$ cp /bin/id ./myid
$ sudo chown root myid
$ ./myid
uid=1000(seed) gid=1000(seed) groups=1000(seed), ...
```

SET-UID

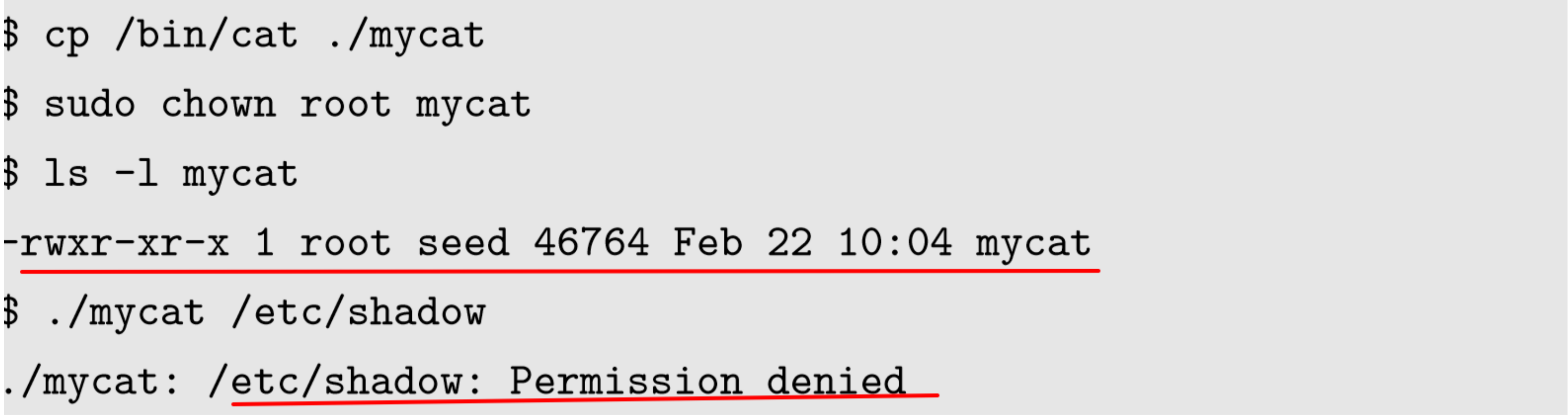
- We can use chmod to **set UID** for a program



```
$ sudo chmod 4755 myid
$ ./myid
uid=1000(seed) gid=1000(seed) euid=0(root) ...
```

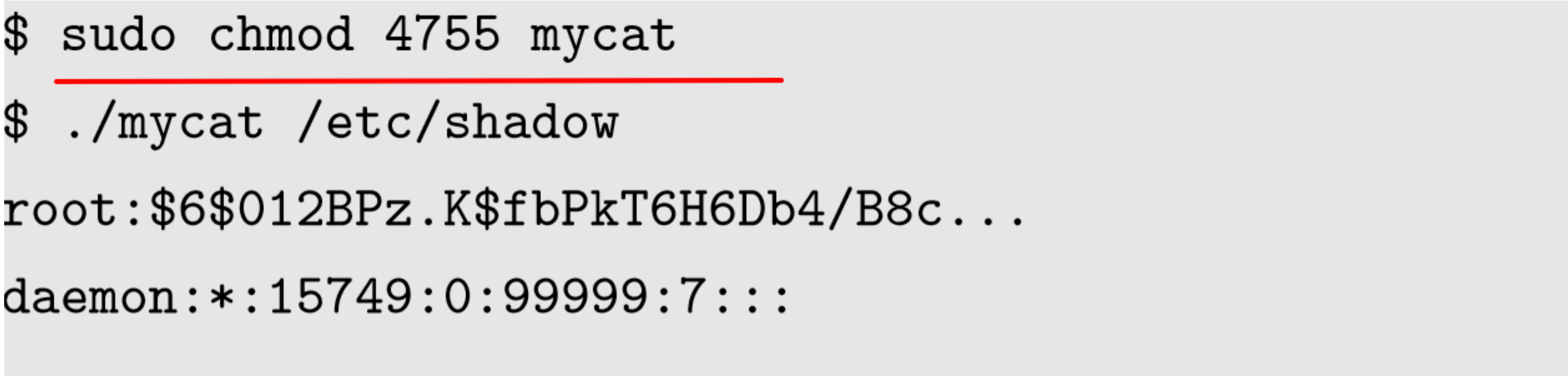
How SET-UID Program Works

- A normal program cannot access /etc/shadow file



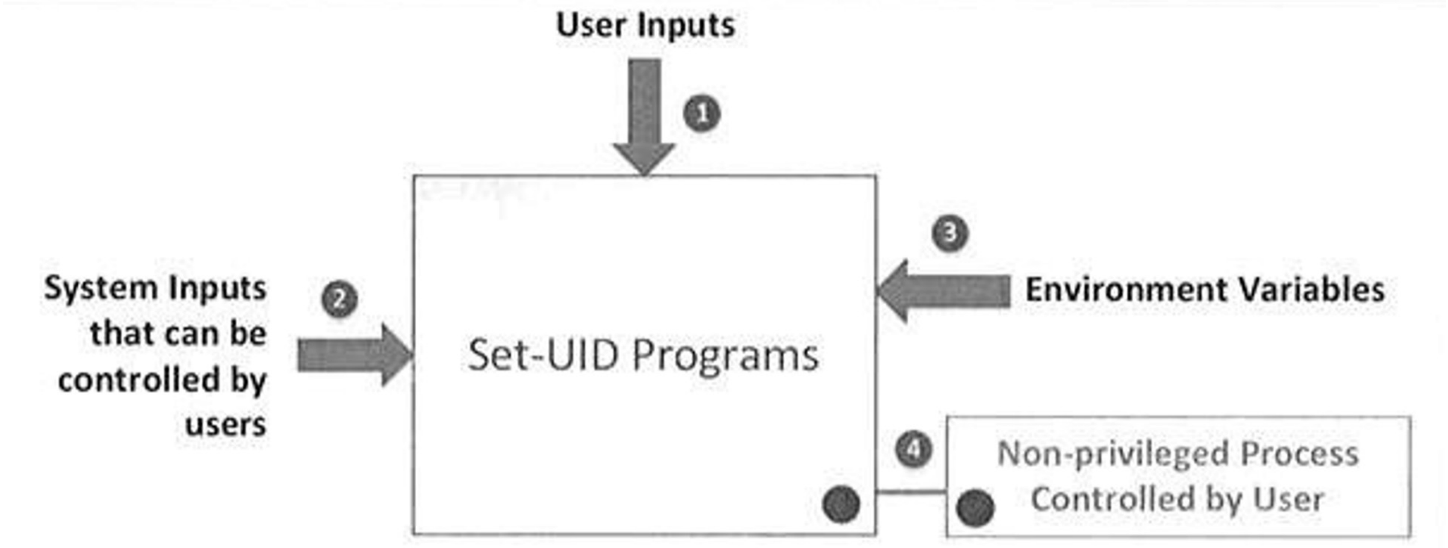
```
$ cp /bin/cat ./mycat
$ sudo chown root mycat
$ ls -l mycat
-rwxr-xr-x 1 root seed 46764 Feb 22 10:04 mycat
$ ./mycat /etc/shadow
./mycat: /etc/shadow: Permission denied
```

- A Set-UID program can



```
$ sudo chmod 4755 mycat
$ ./mycat /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8c...
daemon:*:15749:0:99999:7:::
```

Attack Surface SET-UID Program


User Inputs
1
System Inputs that can be controlled by users
2
Set-UID Programs
3
Environment Variables
4
Non-privileged Process Controlled by User

User Inputs & System Inputs

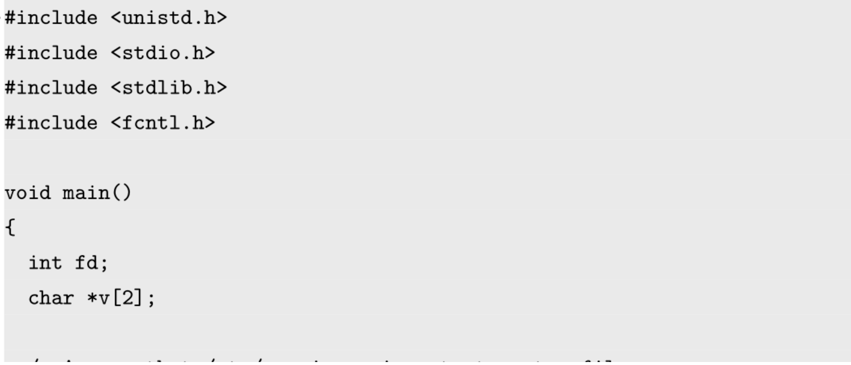
- Buffer overflow of user inputs
    - We have discussed in the previous class
- The program can get inputs from the system
    - Read a file -> symbolic link
        - Make /tmp/xyz -> /etc/shadow : race condition

```
If (!access(file, W_OK)) {
  fopen(file, “a+”) {
  }
}
```

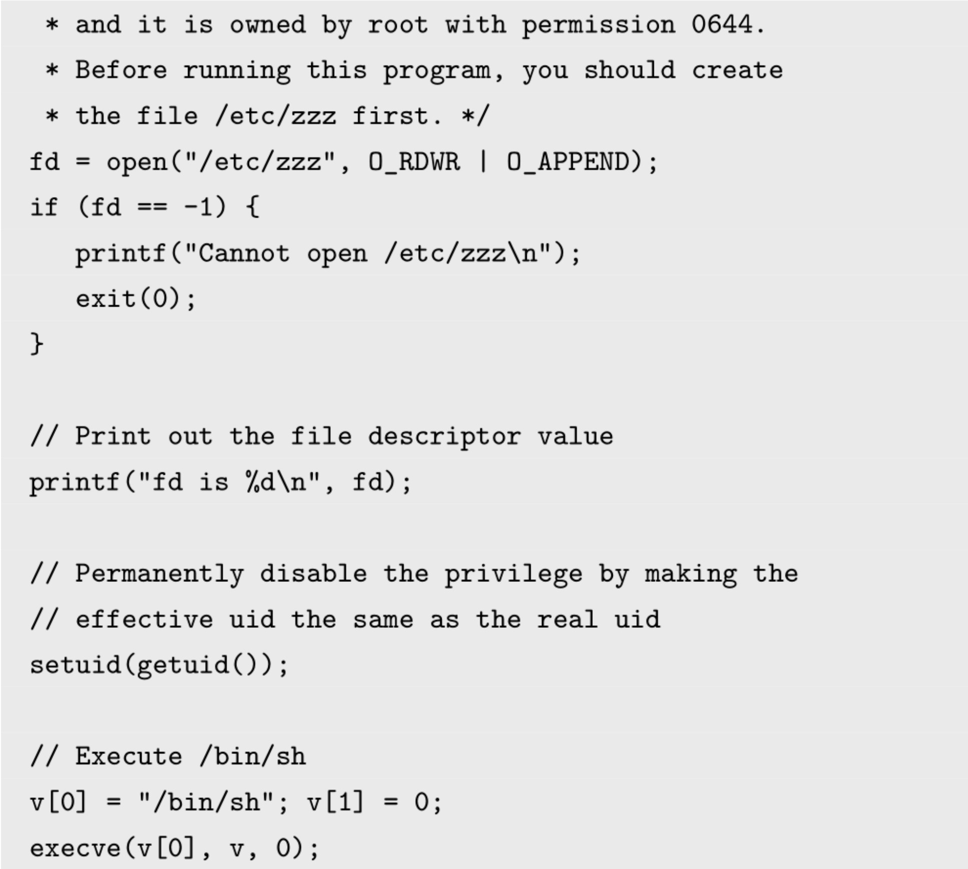
Environments

- We will discuss later

Leak privilege



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

void main()
{
  int fd;
  char *v[2];
```



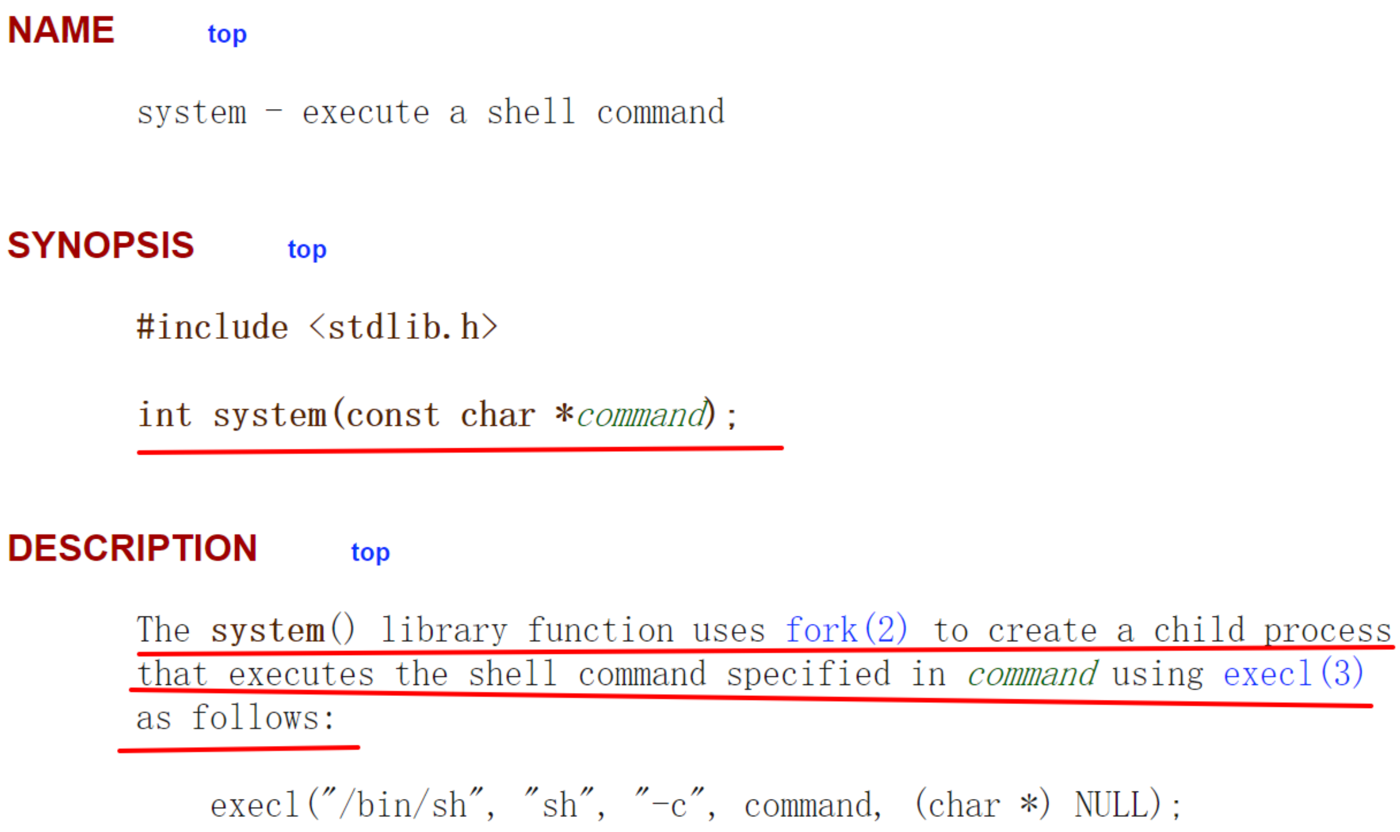
```
   * and it is owned by root with permission 0644.
   * Before running this program, you should create
   * the file /etc/zzz first. */
  fd = open("/etc/zzz", O_RDWR | O_APPEND);
  if (fd == -1) {
     printf("Cannot open /etc/zzz\n");
     exit(0);
  }

  // Print out the file descriptor value
  printf("fd is %d\n", fd);

  // Permanently disable the privilege by making the
  // effective uid the same as the real uid
  setuid(getuid());

  // Execute /bin/sh
  v[0] = "/bin/sh"; v[1] = 0;
  execve(v[0], v, 0);
```

Leak privilege

- /etc/zzz is a file that can only be changed by a privileged program
- However the fd is not closed after setuid. As such this fd can still be used by the program to operate on the underlying file
  - The privilege is only checked when a file is opened.

Invoke Other Programs

- Unsafe way: System() function



NAME top

system - execute a shell command

SYNOPSIS top

```
#include <stdlib.h>

int system(const char *command);
```

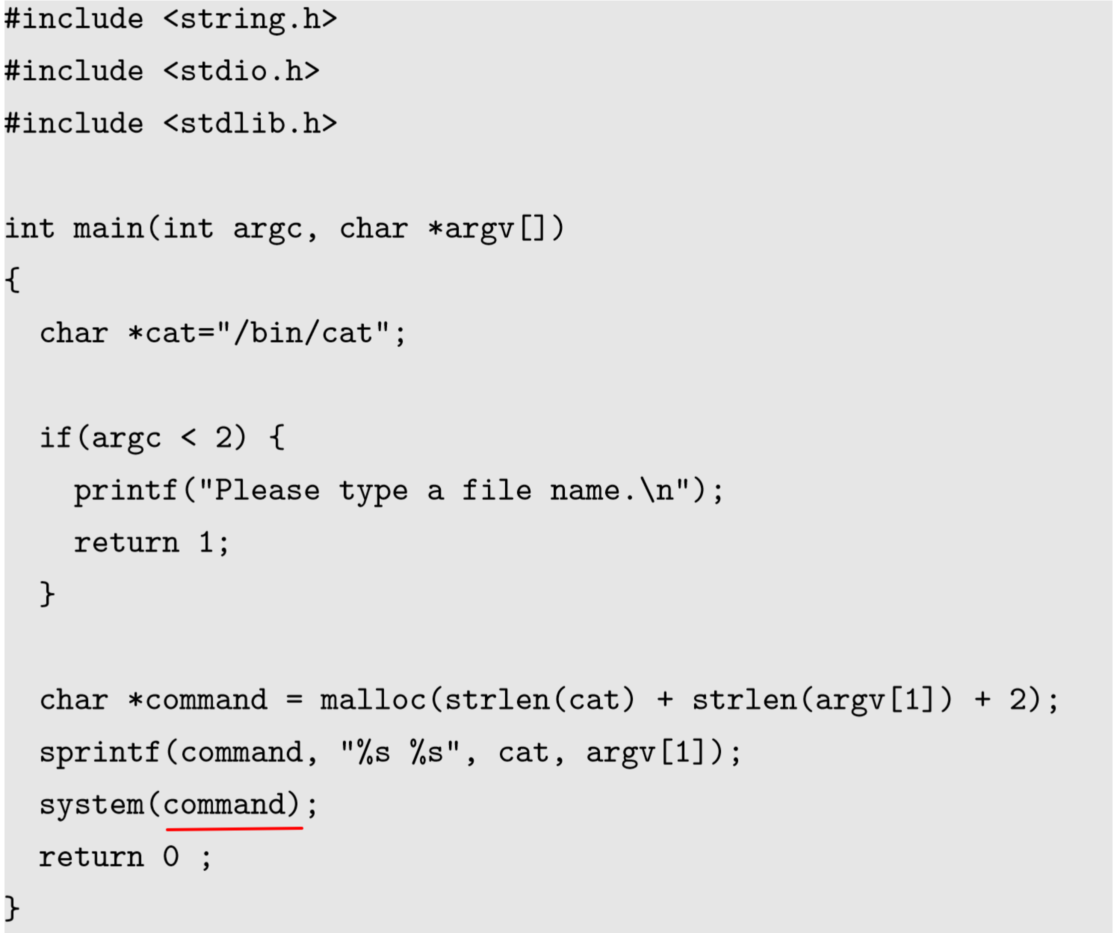
DESCRIPTION top

The **system**() library function uses fork(2) to create a child process that executes the shell command specified in *command* using execl(3) as follows:

```
execl("/bin/sh", "sh", "-c", command, (char *) NULL);
```

Invoke Other Programs

- Unsafe way: System() function
  - Why? System uses /bin/sh to execute commands – *too* powerful



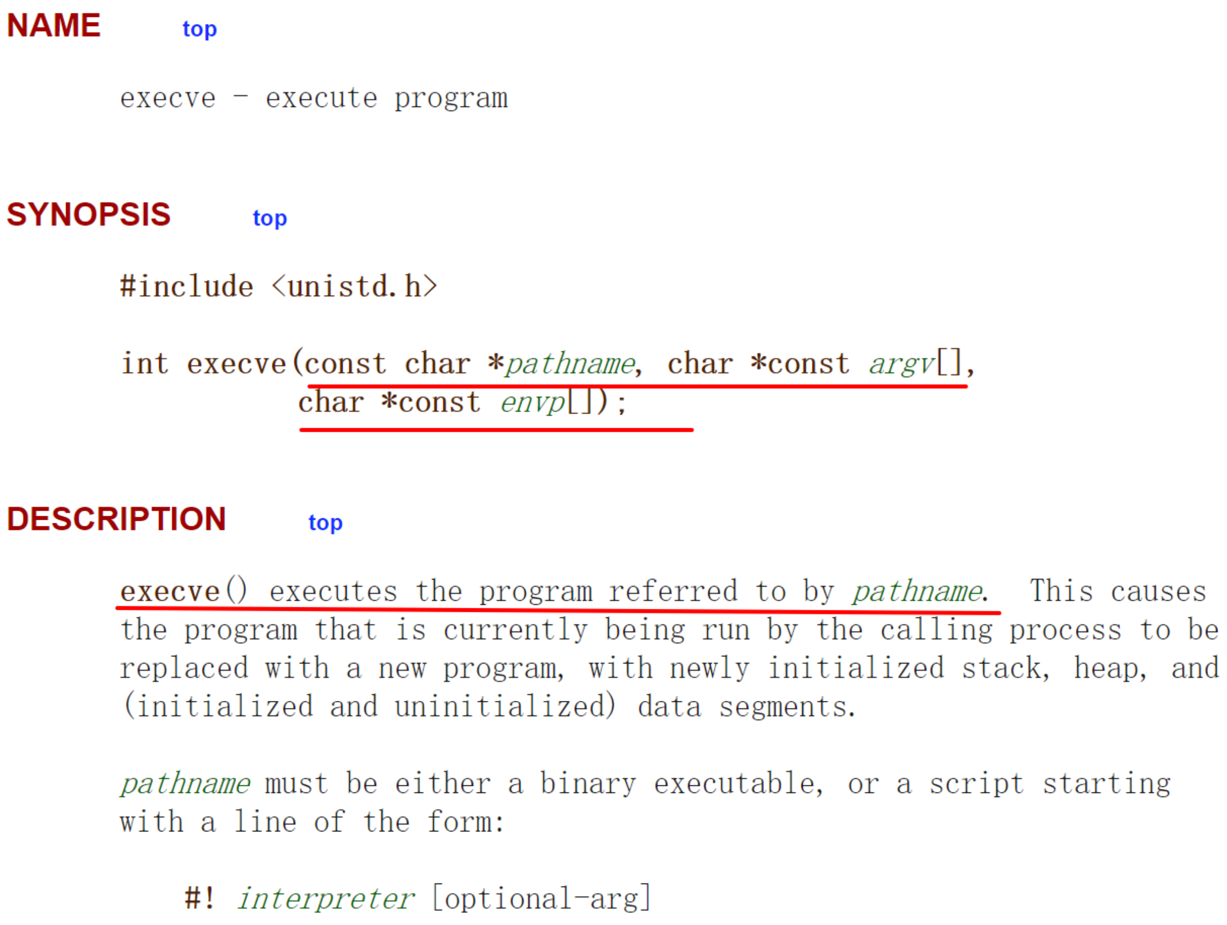
```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  char *cat="/bin/cat";

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  char *command = malloc(strlen(cat) + strlen(argv[1]) + 2);
  sprintf(command, "%s %s", cat, argv[1]);
  system(command);
  return 0 ;
}
```

Invoke Other Programs

```
$ gcc -o catall catall.c
$ sudo chown root catall
$ sudo chmod 4755 catall
$ ls -l catall
-rwsr-xr-x 1 root seed 7275 Feb 23 09:41 catall
$ catall /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8cLWb....
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
sync:*:15749:0:99999:7:::
games:*:15749:0:99999:7:::

$ catall "aa;/bin/sh"
/bin/cat: aa: No such file or directory
#         ← 得到了 root 权限的 shell!
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=0(root), ...
```

Invoke Other Programs

- Safe way: execve() function



NAME top

execve - execute program

SYNOPSIS top

```
#include <unistd.h>

int execve(const char *pathname, char *const argv[],
           char *const envp[]);
```

DESCRIPTION top

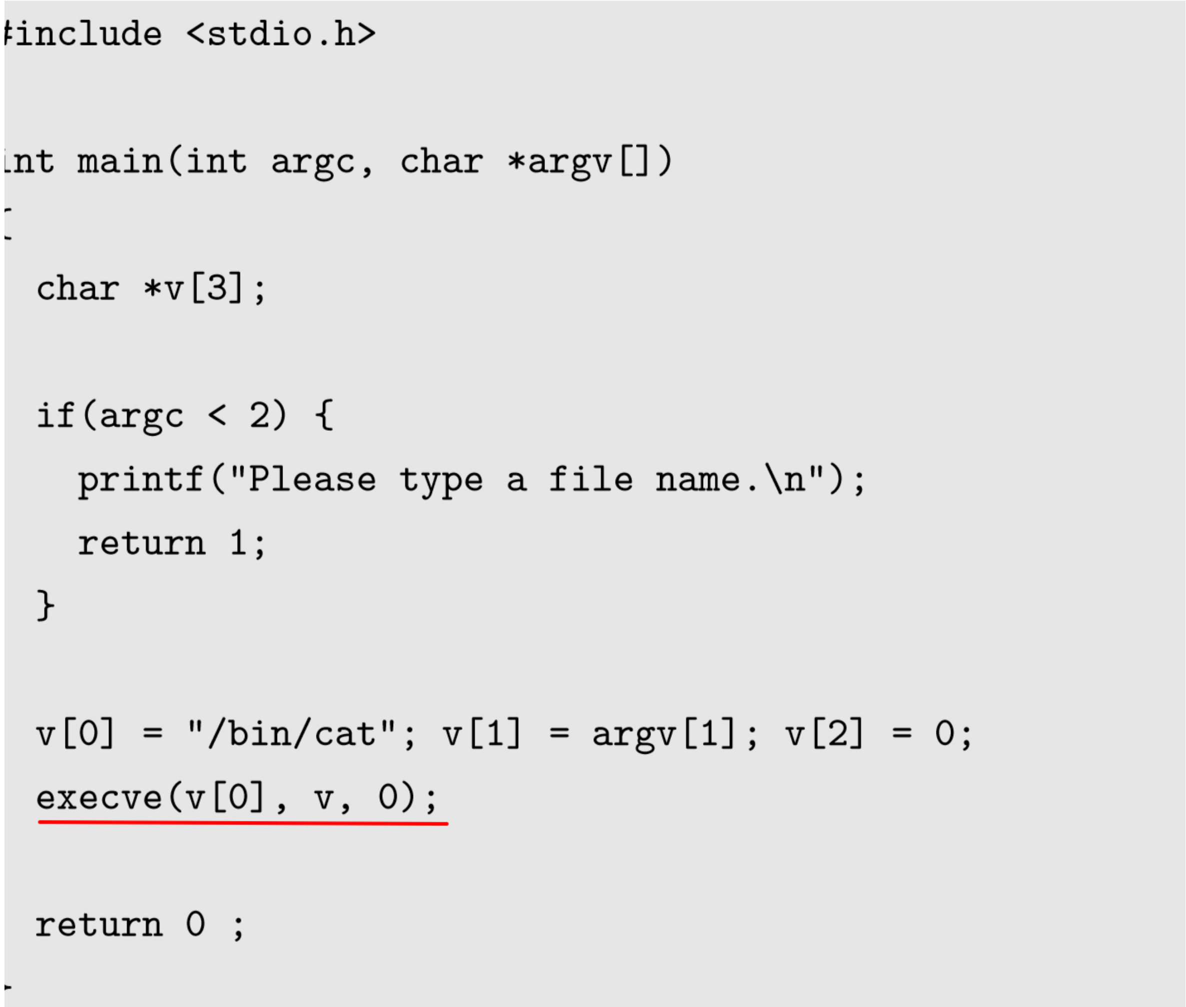
execve() executes the program referred to by *pathname*. This causes the program that is currently being run by the calling process to be replaced with a new program, with newly initialized stack, heap, and (initialized and uninitialized) data segments.

pathname must be either a binary executable, or a script starting with a line of the form:

```
    #! interpreter [optional-arg]
```

Invoke Other Programs

- Safe way: execve() function



```
#include <stdio.h>

int main(int argc, char *argv[])
{
  char *v[3];

  if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
  }

  v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = 0;
  execve(v[0], v, 0);

  return 0 ;
}
```

Invoke Other Programs

- All the inputs will be treated as arguments!

```
$ gcc -o safecatall safecatall.c
$ sudo chown root safecatall
$ sudo chmod 4755 safecatall
$ safecatall /etc/shadow
root:$6$012BPz.K$fbPkT6H6Db4/B8cLWb....
daemon:*:15749:0:99999:7:::
bin:*:15749:0:99999:7:::
sys:*:15749:0:99999:7:::
sync:*:15749:0:99999:7:::
games:*:15749:0:99999:7:::

$ safecatall "aa;/bin/sh"
/bin/cat: aa;/bin/sh: No such file or directory   ← Attack failed!
```