Shell

Process programming

Carmi Merimovich

Tel-Aviv Academic College

December 11, 2018

open/pipe/close system calls

1. `int fd = open(char *name, int flags);`
2. `int pipe(int fd[2]);`
3. `int close(fd);`

The first process

- The kernel sets the initial state to:
  - cwd is "/".
  - No file is open.
- Sets standard input, standard output, and standard error, to the console device.
- Creates a process to run the shell (**sh**).
- Enters an infinite loop of **wait()**'s.

sh main functionality

sh main loop

```
while (read(0, cmd, ...) > 0) {
  if (cmd is internal command)
    executeInternalCmd(cmd);
  else
    forkExternalCmd(cmd);
}
exit();
```

- Internal cmd "cd" causes execution of the `chdir` system call.
- External commands are assumed to be executble files.

sh example: Simple exec

Typing:

```
ls
```

will use the following code, where the parent **sh** executes: and the child **sh** executes:

```
pid = fork();
if (pid == 0) {
  char *argv[] = {"ls", 0};
  exec("ls", argv);
  exit();
}
wait();

pid = fork();
if (pid == 0) {
  char *argv[] = {"ls", 0};
  exec("ls", argv);
  exit();
```

sh example: Simple exec

Typing:

```
ls -l
```

will use the code, where the parent **sh** executes: and the child **sh** executes:

```
pid = fork();
if (pid == 0) {
  char *argv[] = {"ls", "-l",0};
  exec("ls", argv);
  exit();
}
wait();

pid = fork();
if (pid == 0) {
  char *argv[] = {"ls", "-l",0};
  exec("ls", argv);
```

sh example: Output redirection

Typing:

```
ls > a.txt
```

will use the code, where the parent **sh** executes: and the child **sh** executes:

```
pid = fork();
if (pid == 0) {
  close (1);
  open("a.txt", O_CREAT);
  char *argv[] = {"ls", 0};
  exec("ls", argv);
  exit();
}
wait();

pid = fork();
if (pid == 0) {
```

sh example: Output redirection

Typing:

```
ls -l > b.txt
```

will use the code, where the parent **sh** executes: and the child **sh** executes:

```
pid = fork();
if (pid == 0) {
  close (1);
  open("b.txt", O_CREAT);
  char *argv[] = {"ls", "-l", 0};
  exec("ls", argv);
  exit();
}
wait();

pid = fork();
if (pid == 0) {
```

sh example: Input redirection

Typing:

```
sh < b.txt
```

will use the code, where the parent **sh** executes: and the child **sh** executes:

```
pid = fork();
if (pid == 0) {
  close (0);
  open("b.txt", O_RONLY);
  char *argv[] = {"sh", 0};
  exec("sh", argv);
  exit();
}
wait();

pid = fork();
if (pid == 0) {
```

sh example: Pipe

Typing:

```
cat a.bat | sh
```

will use the code: where the parent **sh** executes: the first child **sh** executes: the second child **sh** executes:

```
int p[2];
pipe(p);
pid = fork();
if (pid == 0) {
 close (1);
 dup(p[1]);
 close(p[0]);
 close(p[1]);
 char *argv[] = {"cat", 0};
 exec("cat", argv);
 exit();
}
```

```
pid = fork()
if (pid == 0) {
 close(0);
 dup(p[0]);
 close(p[0]);
 close(p[1]);
 char *argv[] = {"sh", 0};
 exec("sh", argv);
 exit();
}
close(p[0]);
close(p[1]);
wait();
```