Computer Systems II

Executing Processes

Behind the ‘system(…)’ Command

execv
execl

Recall: system()

- Example call:

```
system("mkdir systest");
```

  - **mkdir** is the name of the executable program
  - **systest** is the argument passed to the executable

mkdir.c

```
int main(int argc, char * argv[])
{
  ...
  // create directory called argv[1]
  ...
}
```

3

Unix's execv

- The system call **execv** executes a file, transforming the calling process into a new process. After a successful **execv**, there is no return to the calling process.

```
execv(const char * path, const char * argv[])
```

  - **path** is the full path for the file to be executed
  - **argv** is the argument array, starting with the program name
    - each argument is a null-terminated string
    - the first argument is the name of the program
    - the last entry in argv is NULL

4

system() vs. execv

```
system("mkdir systest");
```

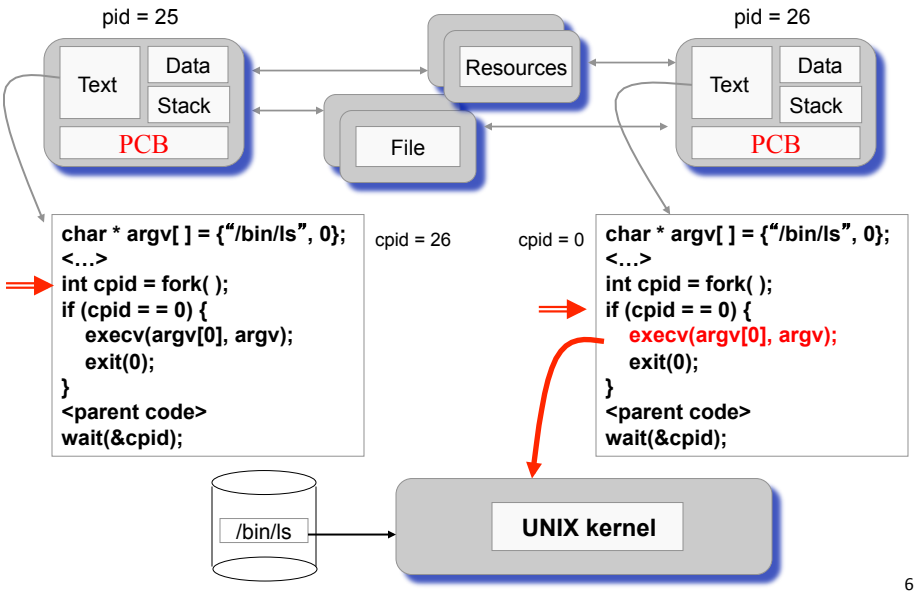
mkdir.c

```
int main(int argc, char * argv[])
{
   ...
   // create directory called argv[1]
   ...
}
```

```
char * argv[] = {"/bin/mkdir", "systest", NULL};
execv(argv[0], argv);
```

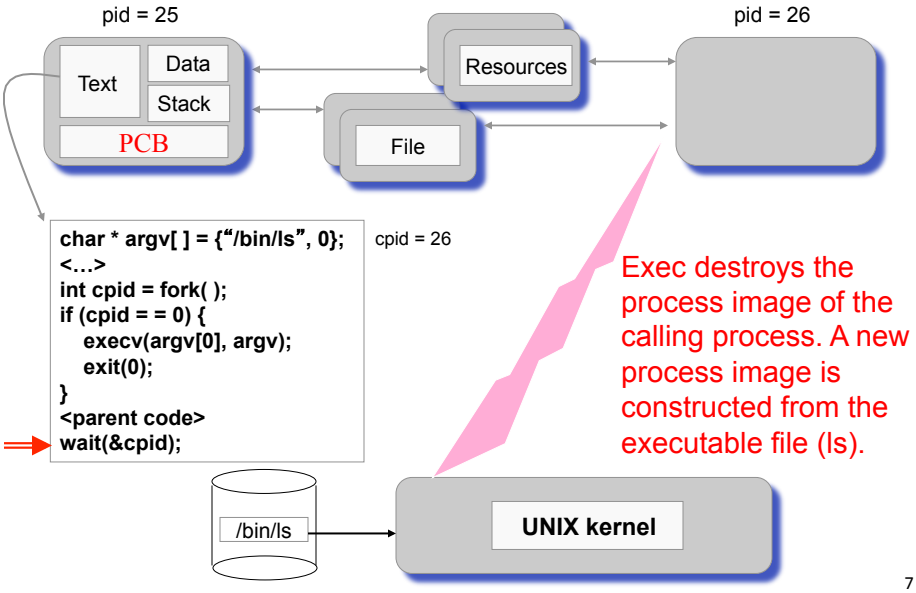
5

How execv Works (1)



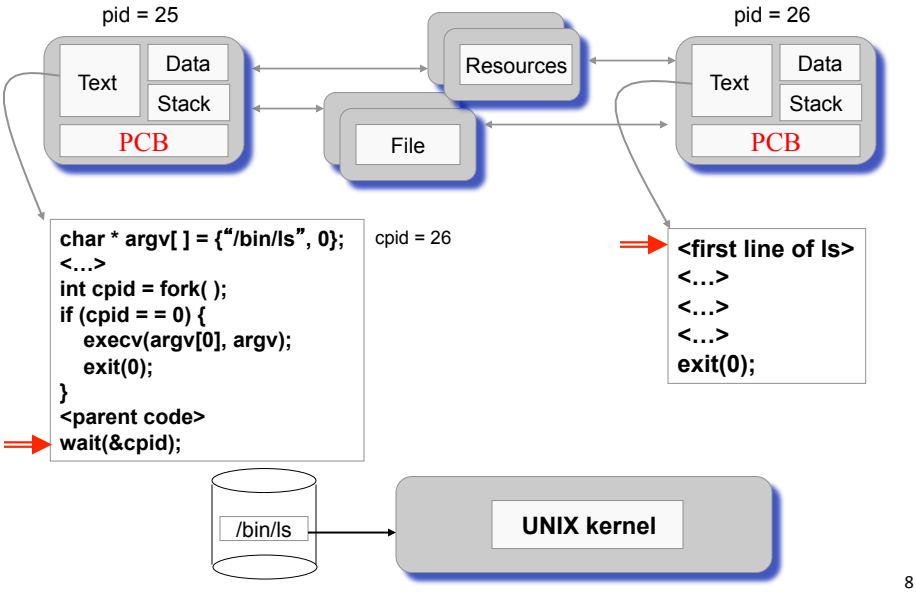
6

How execv Works (2)



7

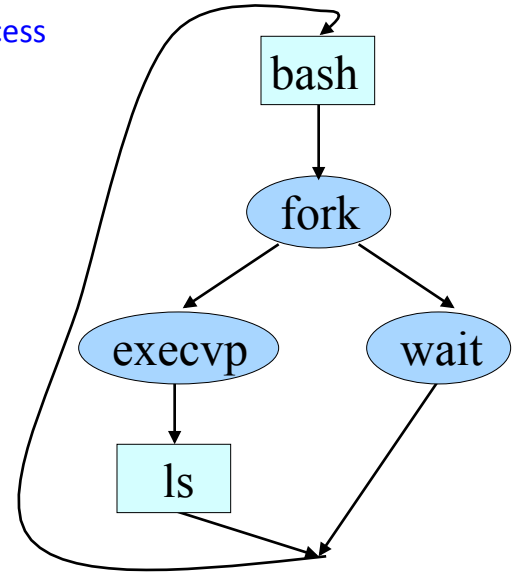
How execv Works (3)



8

Example: Smart Shell

- Shell is the parent process
  - E.g., bash
- Parses command line
  - E.g., "ls –l"
- Invokes child process
  - fork, execvp
- Waits for child
  - wait



9

execv Example

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

char * argv[] = {"/bin/ls", "-l", 0};
int main()
{
   int pid, status;

   if ( (pid = fork() ) < 0 )
   {
        printf("Fork error \n");
        exit(1);
   }
   if(pid == 0) { /* Child executes here */
        execv(argv[0], argv);
        printf("Exec error \n");
        exit(1);
   } else       /* Parent executes here */
        wait(&status);
   printf("Hello there! \n");
   return 0;
}
```

Note the NULL string at the end

10

execv Example – Sample Output

- Sample output:

```
total 282
drwxr-xr-x  2 mdamian  faculty  512 Feb 29 14:02 assignments
-rw-r--r--  1 mdamian  faculty  3404 Feb 29 14:05 index.html
drwxr-xr-x  2 mdamian  faculty  512 Feb 28 15:02 notes

Hello there!
```

11

execl

- Same as execv, but takes the arguments of the new program as a list, not a vector:
- Example:

```
execl("/bin/ls", "/bin/ls", "-l", 0);
```

- This is equivalent to

Note the NULL string at the end

```
char * argv[] = {"/bin/ls", "-l", 0};
execv(argv[0], argv);
```

- execl is mainly used when the number of arguments is known in advance

12

General purpose process creation

- In the parent process:

```
int childPid;
char * const argv[ ] = {…};

main {
   childPid = fork();
   if(childPid == 0)
   {
        // I am child ...
        // Do some cleaning, close files
        execv(argv[0], argv);
   }
   else
   {
        // I am parent ...
        <code for parent process>
      wait(0);
   }
}
```

13

Combined fork/exec/wait

- Common combination of operations
  - Fork to create a new child process
  - Exec to invoke new program in child process
  - Wait in the parent process for the child to complete
- Single call that combines all three
  - int system(const char *cmd);
- Example

```
int main()
{
    system("echo Hello world");
}
```

14

Properties of fork / exec sequence

- In 99% of the time, we call execv(…) after fork()
  - the memory copying during fork() is useless
  - the child process will likely close open files and connections
  - overhead is therefore high
  - might as well combine both in one call (OS/2)

- vfork()
  - a system call that creates a process without creating an identical memory image
  - sometimes called “lightweight” fork
  - child process is understood to call execv() almost immediately

15

Variations of execv

- execv
  - Program arguments passed as an array of strings
- execvp
  - Extension of execv
  - Searches for the program name in the PATH environment
- execl
  - Program arguments passed directly as a list
- execlp
  - Extension of execv
  - Searches for the program name in the PATH environment

16

Summary

- exec(v,vp,l,lp)
  - Does NOT create a new process
  - Loads a new program in the image of the calling process
  - The first argument is the program name (or full path)
  - Program arguments are passed as a vector (v,vp) or list (l,lp)
  - Commonly called by a forked child

- system:
  - combines fork, wait, and exec all in one