CSCE 451/851
Operating Systems Principles

The UNIX Operating System
Processes & IPC

Steve Goddard
goddard@cse.unl.edu

http://www.cse.unl.edu/~goddard/Courses/CSCE451

The UNIX Operating System
History

- The original UNIX
  - » An experimental operating system developed by Ken Thompson & Dennis Ritchie in the late '60s
- Main variants ("Standards")
  - » System V
    - ❖ developed by AT&T, now owned by UNIX International
  - » 4.4 BSD
    - ❖ Open Software Foundation
  - » POSIX
    - ❖ IEEE/ISO
  - » FreeBSD
  - » Linux
- Commercial products
  - » Ultrix, DEC UNIX — DEC
  - » SunOS, Solaris — Sun
  - » HP/UX — Hewlett Packard
  - » AIX — IBM
  - » Xenix — Microsoft
  - » …

The UNIX Operating System

Processes

- A process is created by the **`fork()`** system call
  - » creates a new address space that is a duplicate of the callers

```
main (argc, argv)
 int childpid;
{
 switch (childpid = fork()) {
   case 0:          /* child  */
     child_func();
     exit(0);
   default:         /* parent */
     parent_func();
     while(wait((int *) 0) != childpid);
     exit(0);
   case -1:         /* oops   */
     error("fork:%s\n",sys_errlist[errno]);
  }
}
```

Parent address space

```
childpid = 1
```

Child address space

```
childpid = 0
```

3

The UNIX Operating System

Processes

- Alternatively, processes can be "created" by an **`exec()`**
  - » replaces the memory image of the caller with a new program

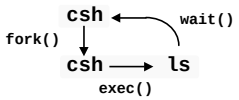
"parent" address space

```
main()
{
 exec( prog)
}
```

"child" address space

```
prog()
{

}
```

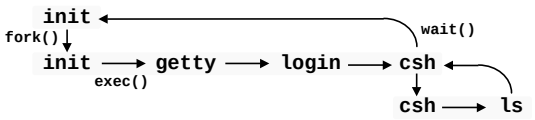
- This is how the shell executes commands
  - » a **`fork()`** followed by an **`exec()`**



4

Processes in UNIX

Example: How users logs in



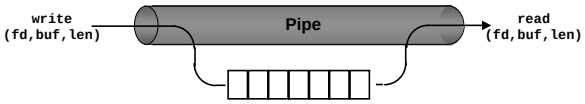
- There exists a "master process" in UNIX: **`init`**
- **`init`** forks a process for each terminal port
- each **`init`** copy execs **`getty`** which prints the login prompt and then reads the login and password
- **`getty`** then execs **`login`** which verifys the login
- **`login`** then execs **`csh`** which forks new processes for each command

5

Processes in UNIX

(Simple) Interprocess communication

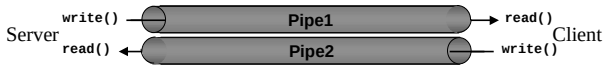
- Like message passing except more general
- Pipes — a shared, in-memory file
  - » a queue of 4K bytes
  - » buffered, asynchronous message passing
    - ❖ blocks reader when queue is empty
    - ❖ blocks writer when queue is full



6

Processes in UNIX

(Simple) Interprocess communication

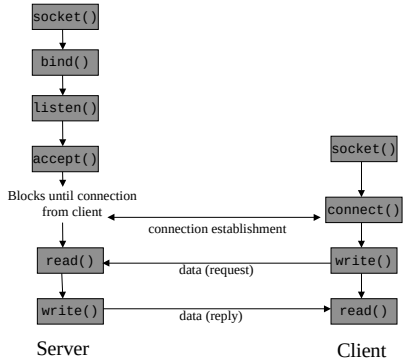



```
main() {
 int pipe1[2], pipe2[2];

 if (pipe(pipe1) == -1  ||  pipe(pipe2) == -1) error(...

 switch (childpid = fork()) {
   case 0:                        /* child */
     close(pipe1[1]);             /* write descriptor for pipe1 */
     close(pipe2[0]);             /* read  descriptor for pipe2 */
     client(pipe1[0],pipe2[1]); /* client program             */
   default :                      /* parent */
     close(pipe1[0]);             /* read  descriptor for pipe1 */
     close(pipe2[1]);             /* write descriptor for pipe2 */
     server(pipe2[0],pipe1[1]); /* server program             */
     while (wait((int *) 0) != childpid);   /* wait for child */
   }
}
```

7

Processes in UNIX

Interprocess communication via Sockets

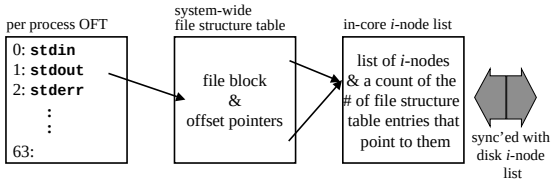


8

The UNIX Operating System

File system

- UNIX maintains an *open file table* for each process which lists each file in use by the process
  - » the OFT is copied when processes are forked

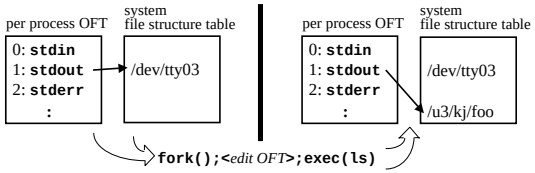


9

The UNIX File System

Open file table examples

- I/O redirection — **`ls > foo`**
  - » just change a pointer in the OFT



- File sharing

10