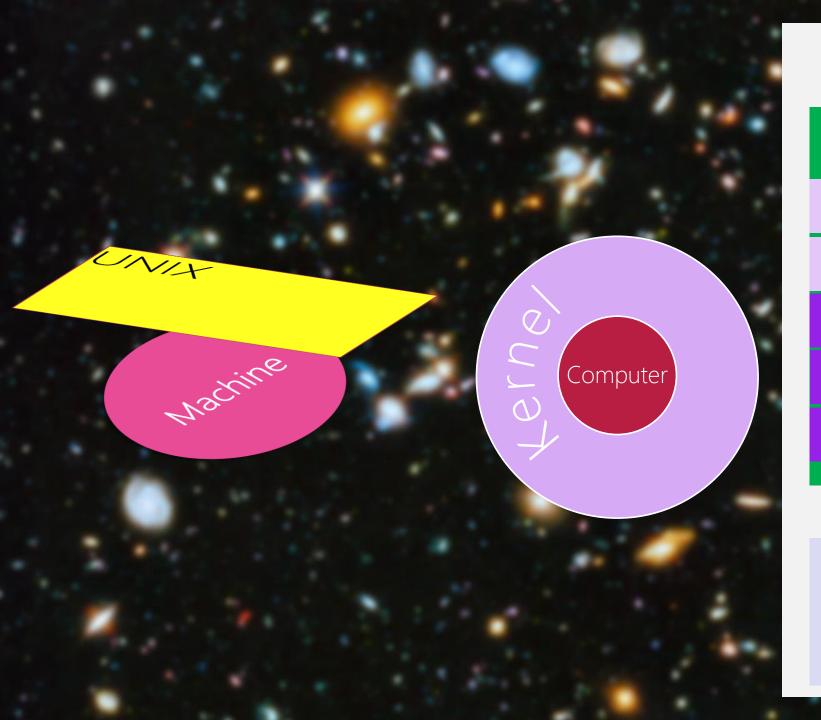


Lab11 & Lec11 This Week



Computer

Memory

Kernel: Device Manager

Kernel: Memory Manager

Kernel: File Manager

Kernel: Network Manager

Kernel: Process Manager

Bus

Processor



So far, ... Technological Services by the Kernel to Application-Level Programs

File Manager (File System): Communicating with Devices and Files

Process Manager (Process Control): Bootstrapping, Multiprocessing, New Process, IPC (Signal, Pipe, ...)

Network Manager: Network IPC, Socket, TCP/IP (UDP, TCP)

Memory Manager: Virtual Memory, Paging, ... → COMP3300: Operating Systems

Device Manager: Device Driver, ... → COMP2660: Microprocessor Programming, COMP????: Assembly Language

Design Patterns for Application-Level Programs

```
fork(), wait(), ... \rightarrow Parent, Child, ...

pipe(), read(), ... \rightarrow Conversational IPC, Producer, Consumer socket(), bind(), ... \rightarrow Sender, Receiver in TCP/IP (UDP) socket(), bind(), ... \rightarrow Client, Server in TCP/IP (TCP) fork(), socket(), ... \rightarrow The Server w/ Children Assigned to Clients
```

You Know the Technology (Material) You Know the Design Patterns

Now, Write a Program

























Problem A

In a process, we have a huge amount of data in memory (e.g., array)

We want to dump it into a file

Design I (Synchronous) open() or creat() → write()

- +: simple, I get the full mark
- -: the program gets stuck in writing to file and cannot do anything else, poor design!

Design II (Asynchronous) open()orcreat() \rightarrow fork() \rightarrow write() by child Is it worth it?

- +: the program continues with other stuff while the child does the writing, I get the full mark plus "good job!"
- -: you need to know what forking is, handle parent/child part of codes.

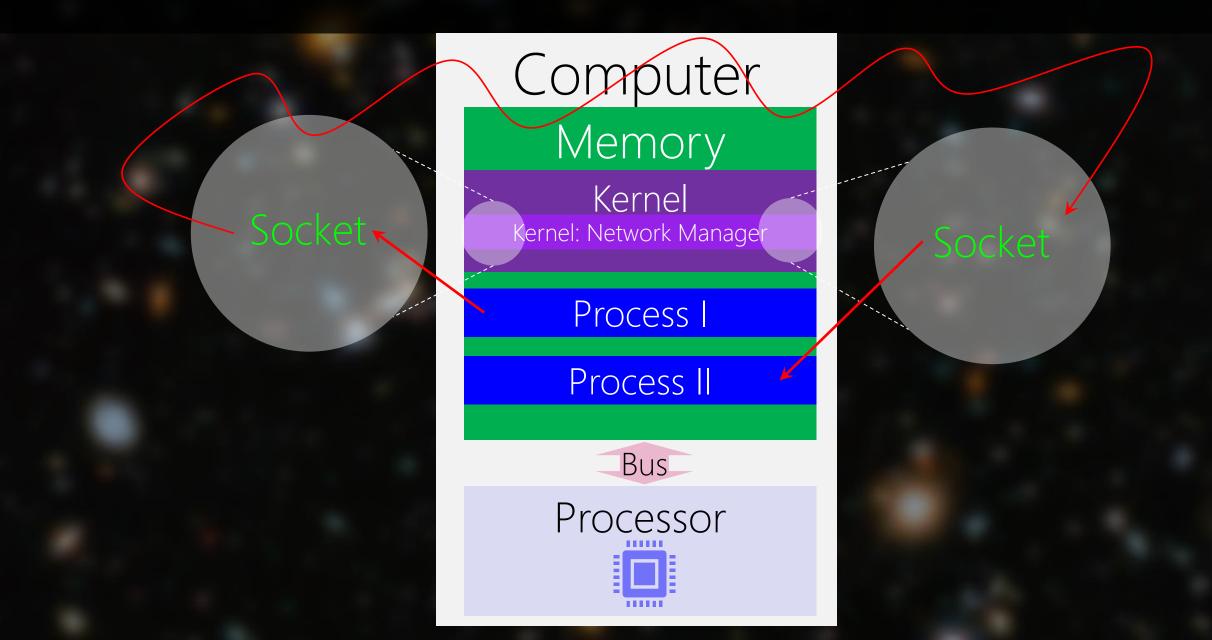
Design III (Asynchronous) open() or creat() $\rightarrow N \times fork() \rightarrow write(part_i)$ by child_i Is it worth it?

- +: the program continues with other stuff while the children do the writing, very fast, I get the full mark plus "WoW!"
- -: you need to know what forking is, handle parent/children part of codes, splitting the data, ...

There is no right or wrong! True or False! There is poor, good, wow designs.

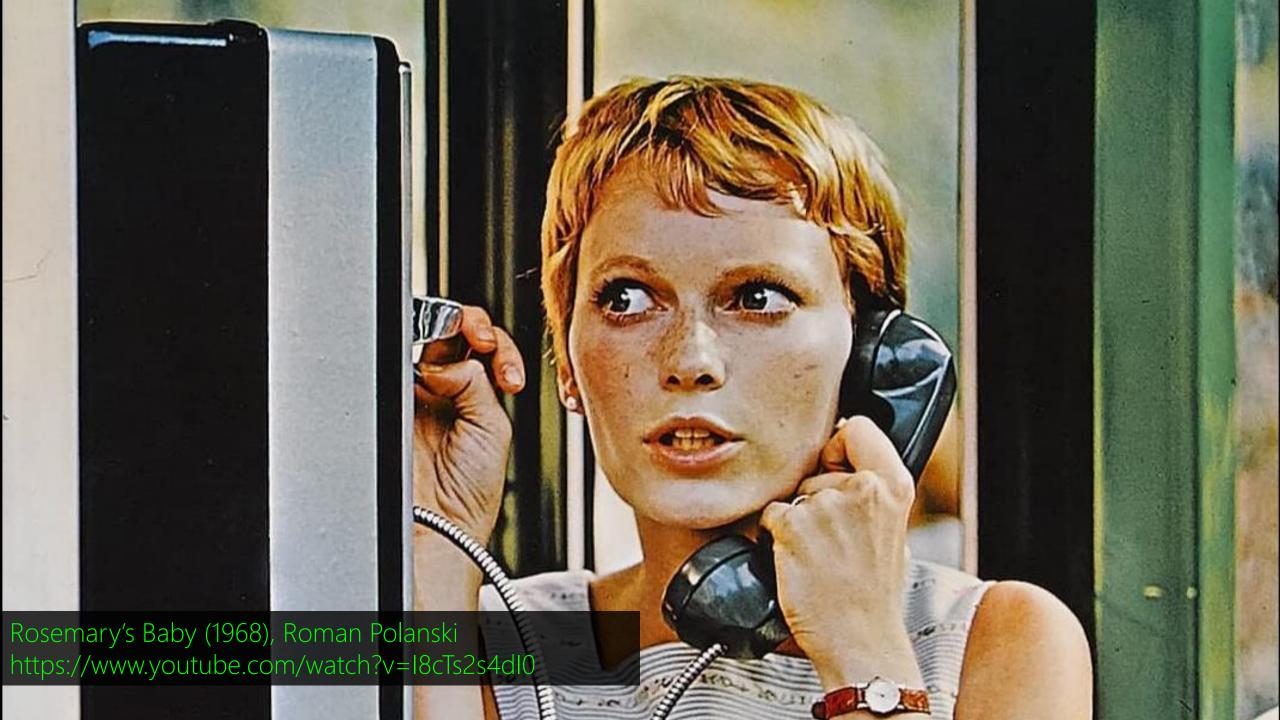
Problem B Network IPC in the Same Computer: an Alternative for IPC Socket vs. Signal, Pipe

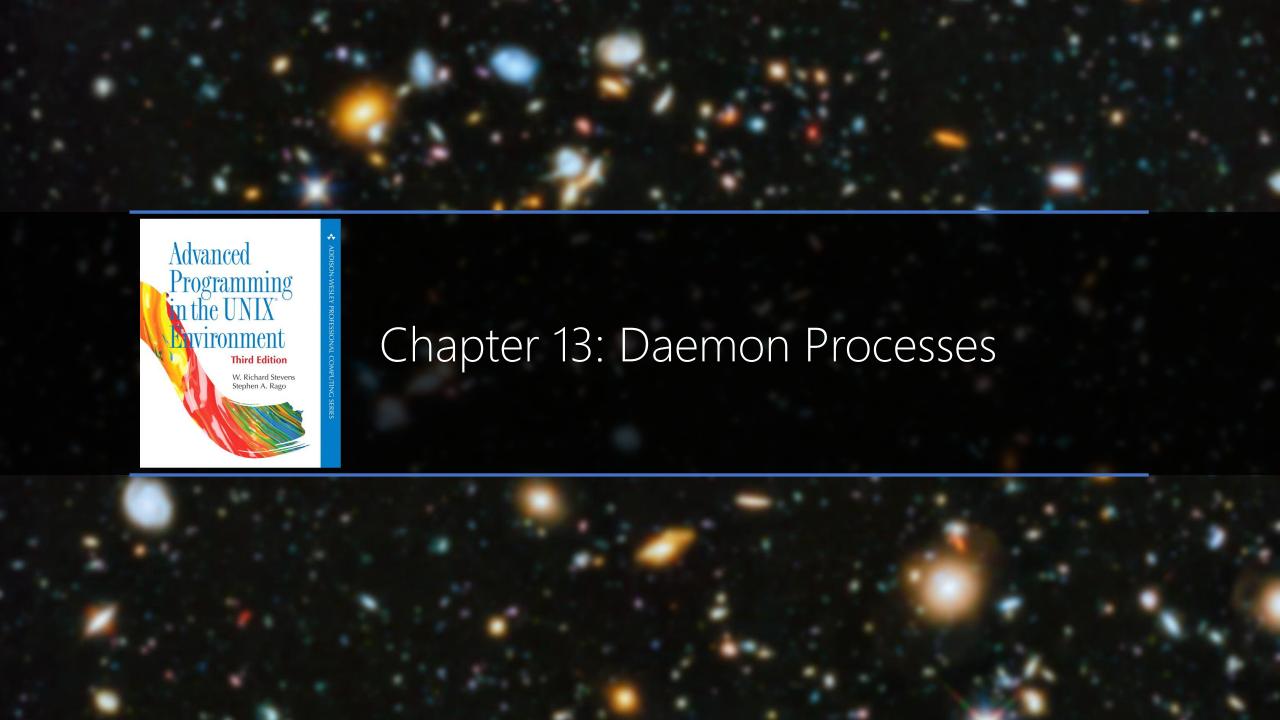
Network IPC on the Same Computer → IPC



Problem Network IPC in the Same Computer: an Alternative for IPC

- +: The parent should not carry the tasks of child, what else?
- -: The parent doesn't control giving birth to the child(ren), no shared file/pipe/mmap, what else?







/'di:mən/ DEE-mən: Evil Spirit

/'deɪmən/ DAY-mən: in Greek mythology's a supernatural being working in the background

https://en.wikipedia.org/wiki/Daemon_(computing)

Daemon Process that lives forever In the background (not visible!) It could be passive or pro-active.

Daemon lives forever It resides in memory till the computer is off Memory-resident process

Daemon lives in the background (not visible, like a demon)

It does not have direct interaction with user
It does not have input/output terminal
But it can have indirect interaction (e.g., files, devices, ...)

Daemon can be passive

Waits for other processes to initiate a communication Then it provides a service

Daemon can be pro-active It does many tasks E.g., routinely report/log the processor activity in a file Keylogger!!

An obvious example of a Daemon process?

The Kernel (Why?)

Another example of a Daemon process?

hfani@alpha:~\$./server socket has created for The Server with sd:3 The Server bound to the address:port = 861065097:51742

hello back

hello back

hfani@charlie:~\$

hfani@charlie:~\$./client

socket has created for the client with sd:3

client is connected to The Server at address:port = 894619529:51742

There is a small problem tho! What?

```
hfani@alpha:~$ ./server
socket has created for The Server with sd:3
The Server bound to the address:port = 861065097:51742
```

It is not in the background!

It's interacting with user. The user cannot use this terminal anymore.

hfani@alpha:~\$./server socket has created for The Server with sd:3 The Server bound to the address:port = 861065097:51742

Putting it in the background, alive, responding to clients
But not visible

```
hfani@alpha:~$ ./server
socket has created for The Server with sd:3
The Server bound to the address:port = 861065097:51742
```

Daemonize Putting it in the background, alive, but not visible How?

Think about it as a designer, considering all previous technologies, patterns,



```
int main(void) {
        int domain = AF INET;//Network Protocol: TCP/IP
        int type = SOCK STREAM;//Connection-Oriented
        int protocol = 0;//Default transport: TCP for Internet connection-oriented
        int server_sd;//socket descriptor ~= file descriptor
        server sd = socket(domain, type, protocol);
        if (server sd == -1) {
                printf("error in creating socket for The Server!\n");
        else
                printf("socket has created for The Server with sd:%d\n", server_sd);
        struct in addr server sin address;
       server_sin_address.s_addr = inet_addr("137.207.82.51");//nslookup `hostname`
int server_sin_port = htons(7882);//larger_than_1024
        struct sockaddr in server sin;
        server sin.sin family = domain;
        server sin.sin addr = server sin address;
        server_sin.sin_port = server_sin_port;
        int result = bind(server sd, (struct sockaddr *) &server sin, sizeof(server sin));
                             or in binding The Server to the address:port = %d:%d\n", server_sin.sin_addr, server_sin.sin_port);
        else
                printf("The Server bound to the address:port = %d:%d\n", server sin.sin addr, server sin.sin port);
        if (listen(server_sd, 5) < 0) {</pre>
                perror("
                exit(1);
        struct sockaddr_in client_sin;//I want to know who send the message
        int client_sin_len;
        while(1)
                result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
                if (result == -1) {
                                      in opening the request from client %d:%d !\n", client_sin.sin_addr, client_sin.sin_port);
                else
                        printf("The Server opened the request from client %d:%d\n", client_sin.sin_addr, client_sin.sin_port);
                char msg[10];
                recv(result, msg, 10, 0);
                printf("%s\n", msg);
                send(result, "hello back", 11, 0);
```

The Server (server.c)

```
int main (void) {
        int domain = AF INET;/
        int type = SOCK STREAM;//Connection-Oriented
        int protocol = 0;//Default transport: TCP for Internet connection-oriented
        int server_sd;//socket descriptor ~= file descriptor
        server sd = socket(domain, type, protocol);
        if (server sd == -1) {
                printf("error in creating socket for The Server!\n");
        else
                printf("socket has created for The Server with sd:%d\n", server_sd);
        struct in addr server sin address;
        server_sin_address.s_addr = inet_addr("137.207.82.51");//nslookup `hostname'
int server_sin_port = htons(7882);//larger_than 1024
        struct sockaddr in server sin;
        server sin.sin family = domain;
        server sin.sin addr = server sin address;
        server_sin.sin_port = server_sin_port;
        int result = bind(server sd, (struct sockaddr *) &server sin, sizeof(server sin));
                printf("error in binding The Server to the address:port = %d:%d\n", server_sin.sin_addr, server_sin.sin_port);
        else
                printf("The Server bound to the address:port = %d:%d\n", server sin.sin addr, server sin.sin port);
        if (listen(server_sd, 5) < 0) {</pre>
                perror ("The
                exit(1);
        struct sockaddr_in client_sin;//I want to know who send the message
        int client_sin_len;
                result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
                if (result == -1) {
                                    or in opening the request from client %d:%d !\n", client_sin.sin_addr, client_sin.sin_port);
                else
                        printf("The Server opened the request from client %d:%d\n", client_sin.sin_addr, client_sin.sin_port);
                char msg[10];
                recv(result, msg, 10, 0);
                printf("%s\n", msg);
                send(result, "hello back", 11, 0);
```

The Server (server.c)

Daemonize {process name} → {process name} d

Convention for naming of programs to be run as a daemon

```
int main (void) {
                int child pid = fork();
                if (child pid > 0)
                                 exit(0);
                else if (child pid == -1) {
                                                            t create a child for The Server!\n");
                                 exit(1);
                        server_sin.sin_family = domain;
server_sin.sin_addr = server_sin_address;
                          int result = bind(server sd. (struct sockaddr *) &server sin. sizeof(server sin));
                         struct sockaddr in client sin; //I want to know who send the messa
                               result = accept(server_sd, (struct sockaddr *) &client_sin, &client_sin_len);
                              recv(result, msg, 10, 0);
printf("%s\n", msg);
send(result, "hello back", 11, 0);
```

The Server as a daemon (serverd.c)

- 1) Wrap it for a child
- 2) Parent exits -
- 3) Orphan child lives forever (Adopted by the kernel

```
hfani@alpha:~$ ./serverd
socket has created for The Server with sd:3
The Server bound to the address:port = 861065097:51742
hfani@alpha:~$
```

It returns to the shell, ready for other commands or processes

```
hfani@alpha:~$ ./serverd
socket has created for The Server with sd:3
The Server bound to the address:port = 861065097:51742
hfani@alpha:~$
hfani@alpha:~$ ps
    PID TTY
                     TIME CMD
1568863 pts/6 00:00:00 bash
1586249 pts/6
                00:00:00 serverd
1586307 pts/6
                00:00:00 ps
```

It returns to the shell, ready for other commands or processes

But it is in memory, passive, waiting for clients

```
hfani@alpha:~$ ./client
socket has created for the client with sd:3
client is connected to The Server at address:port = 861065097:51742
The Server opened the request from client 21895:15486
hello
hello back
```

These lines are not printed by the client program They are printed by The Server!

Because both are using same standard output fd=1

Daemonize Print out (log) to another Device/File

open (./serverd.log), write()

Daemonize

- 1) Create a child and give all task to the child
- 2) Exit the parent
- 3) Any printing to a log file
- 4) By convention, add "d" to the program file name
- 5) Other advanced steps (Advanced System Programming)

Is there any other way to daemonize?

How to end a daemon?

