CS 340
Computer Systems

#12: IPC and Networking

Sep. 28, 2023 · G Carl Evans

Threads vs. Processes

Up until now, we've discussed **threads** -- the fundamental unit of computation -- and we know they're organized into **processes**.

- Threads within a process share nearly **all** resources (exceptions are few, like the PC and their stack frames).
  AND
- Processes are almost ________________ from other processes.

	Threads	Processes
Creation		
Overhead		
Context Switching		
Virtual Memory		

Case Study: Chrome

-

Inter-Process Communication (IPC)

IPC is the broad terminology for all technologies that facilitate real-time communication between processes.

Approach #1: __________________________

Using a pipe within a terminal:

```
$ ps -aux | grep waf
```

Creating pipes in C:

```
int pipe(int pipefd[2]);
```

Approach #2: _______________________

Approach #3: _______________________

Sending a signal within a terminal:

```
$ kill -TERM <pid>
```

Listing all available signals:

```
$ kill -l
```

Sending a signal in C:

```
int kill(pid_t pid, int sig);
```

Approach 4: _______________________

Allocating shared memory in C ("malloc for shared memory"):

```
void *mmap(void *addr, size_t length, int prot, int
flags, int fd, off_t offset);
```

Approach 5: _______________________

Functions in C:

```
mqd_t mq_open(const char *name, int oflag);
int mq_send(mqd_t mqdes, const char *msg_ptr,
            size_t msg_len, unsigned int msg_prio);
ssize_t mq_receive(mqd_t mqdes, char *msg_ptr,
            size_t msg_len, unsigned int *msg_prio);
int mq_close(mqd_t mqdes);
```

Approach 6: _______________________

Approach 7: ____________________

Creating a new socket interface, returns a **file descriptor**:

```
int socket(int domain, int type, int protocol);
```

Binding a socket interface to an address and port:

```
int bind(int sockfd, const struct sockaddr *addr,  socklen_t addrlen);
```

Connecting to a remote socket:

```
int connect(int sockfd, const struct sockaddr *addr,
            socklen_t addrlen);
```

Begin listening for a remote socket connection:

```
int listen(int sockfd, int backlog);
```

Start a new socket channel with a remote host:

```
int accept(int sockfd, struct sockaddr *restrict addr,
           socklen_t *restrict addrlen);
```

High Level Overview of Sockets

At the core of socket-based IPC, you have a ____________________ coming from a "remote host".

- ___________:
- ___________:
- Port Number:

Server sp23-cs340-adm.cs.illinois.edu:34000	Client

11/socket.c

```
int main() {
  // socket:
  int sockfd = socket(AF_INET, SOCK_STREAM, 0);
...
  // bind:
  if ( bind(sockfd,
            (const struct sockaddr *)&server_addr,
            sizeof(server_addr)) != 0) {   [...]
...
  // listen:
  if (listen(sockfd, 10) != 0) {   [...]
...
  // continue to accept new connections forever:
  while (1) {
    // accept:
    int *fd = malloc(sizeof(int));
    *fd = accept(sockfd, (struct sockaddr *)&client_address,
                 &client_addr_len);
...
    pthread_create(&tid, NULL, client_communication_thread, fd);
  }
}
```

```
void *client_communication_thread(void *vptr_fd) {
  int fd = *((int *)vptr_fd);
  char buffer[4096];

  while (1) {
    // recv message:
    ssize_t len = recv(fd, buffer, 4096, 0);
    buffer[len] = '\0';

    printf("[%d]:   recv(): ", fd);
...
    // send response:
    sprintf(buffer, "Your %ld bytes were received, thank you for
sending them!\n", len);
    send(fd, buffer, strlen(buffer), 0);
  }
```

Simple Socket Communication: `telnet`

The Linux utility **`telnet`** provides simple socket communications by sending all data you enter directly over the socket:

```
$ telnet sp23-cs340-adm.cs.illinois.edu 34000
```

(To exit, press **`Ctrl+]`** to go into command mode; then type **`quit`**.)