IoT: Client Devices

Daemons in C

OS Services

Kernel Services, OS Services

- Daemons have access to Kernel and OS services
- Networking, system logging, processes, filesystem

Things daemons don't have

- A console!
- A user!
- A home directory!
- User interaction!

Console Mangement

Daemons don't have a console

- So what do stderr, stdout, and stdin mean?
- Nothing - you need to manage

Close standard file descriptors

```
close(STDIN_FILENO);
close(STDOUT_FILENO);
close(STDERR_FILENO);
```

Signal Management

Controlling interactive programs

- Unixes give ctrl-c, ctrl-z, etc.
- Not daemons!
- Remember, these just submit *signals to processes* (see: **man kill**)

Signal Management

```
signal(SIGKILL, _signal_handler);
signal(SIGTERM, _signal_handler);
signal(SIGHUP,  _signal_handler);
```

Logging

No console -> no standard output

- Many programs will log to STDERR, or STDOUT
- But we closed them!

Syslog

- Syslog is a systems-wide logger
- /var/log/messages or /var/log/syslog

Open a log and log to it

- openlog(.), syslog(.), closelog()
- see: **man syslog**

Working Directory

No User -> no default working directory

- We need a working directory
- We do handle files

Moving and setting a working directory

```
chdir(WORKING_DIR);
```

File Creation

We do create files

- Need to set default permissions on created files
- Usually files only read by privileged users
- In our case, better to leave open

Setting default permissions

```
umask(S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH)
```

Sessions

A *Session* has one or more *process groups*

- The first process in the session is default session leader

A *process group* has one or more *Processes*

- The group leader and child processes

Think of sessions as *Terminal Sessions*

`setsid()`

Forking

Don't lock up spawning process

- Daemons need their own dedicated processes
- forking a process creates a copy of the program in another process
  - Parent process gets PID; child gets 0; error is negative

USING FORK()

```
PID_T PID = FORK();
IF (PID > 0) EXIT(0);
IF (PID < 0) EXIT(1);
```