IPC II: fifos and mmap

CS 241

Oct. 30, 2013

Pipes and fifos

- On Monday:
  - Pipes are a one-directional stream of data used for IPC.
  - While waiting for the writer to write to the pipe, read() is a blocking call.

- Generic structure:

```
int fds[2];
pipe(fds);
int read_fd = fds[0], write_fd = fds[1];
pid_t pid = fork();
if (pid == 0) { /* child has one end of the pipe */ }
else if (pid > 0) { /* parent has the other end */ }
```

Inter-Process Communications

- Pipes require a common process in the process chain to create the pipe for communication:

fifo

- A **fifo** (sometimes called a “named pipe”) is the solution to the common-parent problem!

- A fifo is:
  -
  -

Using a fifo

- Create a fifo at the command line:

```
mkfifo name
```

```
void main() {

}
```

```
void main() {

}
```

mmap

- The third form of IPC is **mmap**: a memory map.

- An mmap is a shared region of memory between multiple processes.
  - Two ways to create this memory:
    -
    -

File-backed mmap

- A file-backed mmap maps a file on disk into memory.
  -
  -
- The mmap() call is used to set this up:

```
mmap(NULL, size_t length, int prot,
     int flags, int fd, off_t offset);
```

mmap

- Two ways to use a file-backed mmap:
  - **MAP_SHARED**: Share this mapping. Updates to the mapping are visible to other processes that map this file, and are carried through to the underlying file. The file may not actually be updated until msync(2) or munmap() is called.
  - **MAP_PRIVATE**: Create a private copy-on-write mapping. Updates to the mapping are not visible to other processes mapping the same file, and are not carried through to the underlying file.

```
void main() {

}
```

```
void main() {

}
```

Anonymous mmap

- An anonymous mmap is effectively a malloc() that survives a fork().

- Same system call, with three differences:
  - **offset** field is ignored
  - **fd** must be -1
  - **flags** must contain MAP_ANONYMOUS

```
void main() {

}
```

```
void main() {

}
```