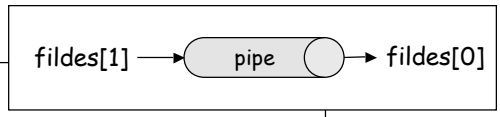
UNIX Special Files

- Pipes
- Named Pipes
- Special Files and Devices

- Reading: R&R, Ch 6

Pipes




```
#include <unistd.h>

int pipe(int fildes[2]);
/* fildes[0] for reading, fildes[1] for writing */
```

```
#include <stio.h>
#include <unistd.h>

int main(int argc, char ** argv) {
  int fd[2]; char result[MAX_LENGTH];
  pipe(fd);
  if (fork())
    fprintf(fd[1], "I am writing into the pipe\n");
  else {
    fscanf("%s", result);
    printf("I read <%s> from the pipe.\n", result);
  }
}
```

Use Pipes for Synchronization

Example: Barrier Synchronization using Pipes:

```
int main(int argc, char ** argv) {
    int fd[2];
    n = atoi(argv[1]);
    pipe(fd);
    for (int i=1; i<n; i++)
        /* start children */
        if ((childpid = fork()) <= 0) break;
    if (childpid > 0) {
        /* parent writes synch characters into pipe */
        for (int j=0; j<n; j++)  write(fd[1], "g", 1);
    }
    read(fd[0], buf, 1); /* everybody synchronizes here */
}
```

Named Pipes (FIFOs)

```
#include <sys/stat.h>

int mkfifo(const char *path, mode_t mode)
```

```
#define MY_PERMS (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

if (mkfifo("myfifo", MY_PERMISSIONS) == -1)
    perror("Failed to create myfifo");

/* use the named pipe. */

/* delete fifo */
if (unlink("myfifo") == -1)
    perror("Failed to remove myfifo");
```

Example: Client-Server Interaction

```
/* SERVER PROGRAM */
int main(int argc, char ** argv) {
  int requestfd;
  string fifoname = argv[1];
  /* -- CREATE NAMED PIPE */
  mkfifo(fifoname, FIFO_PERMS);
  /* -- OPEN READ/WRITE ENDPOINT TO PIPE */
  requestfd = open(fifoname, O_RDWR));
  /* -- WRITE INCOMING INFO TO STDOUT. */
  copy_file(requestfd, STDOUT_FILENO);
}
```

Definition:

A **server** is a subsystem that provides some type of service to *a priori* unknown clients.

```
/* CLIENT PROGRAM */
int main(int argc, char ** argv) {
  int requestfd;
  string fifoname = argv[1];
  /* -- OPEN READ/WRITE ENDPOINT TO PIPE */
  requestfd = open(fifoname, O_WRONLY));
  log_info = generate_some_log_info;
  /* -- SEND LOG INFO TO SERVER PROGRAM */
  r_write(requestfd, STDOUT_FILENO);
  r_close(requestfd);
}
```

Device Control Example: /dev/audio

Device access through file system:

```
cat sample.au > /dev/audio
```

```
/* open audio device just like a file */
int open_audio(void) {
  while (((audio_fd = open("/dev/audio", O_RWR)) == -1)
         && (errno == EINTR));
  if (audio_fd == -1) return -1;
  return 0;
}

/* use ioctl to access and manipulate device parameters */
uint_t get_sample_rate(void) {
  audio_info_t myaudio;
  ioctl(audio_fd, AUDIO_GETINFO, &myaudio);
  return myaudio.play.sample_rate;
}
```