# Named Pipes

**Benjamin Brewster**

Slides adapted from Jon Herlocker, OSU

# FIFOs

- FIFO = First-In, First-out
  - Also called a "named pipe"
- Essentially, a persistent pipe
  - Represented by a special file in the file system
- Any process can open a FIFO
  - If the process has sufficient permissions
    - Permissions work just like a file
  - FIFO is opened like a file - using open()
- Once opened - works just like a pipe

# Looks like, smells like

- FIFOs exist as special files in the file system
- You can create a FIFO, with the mkfifo() system call, or with the mknod or mkfifo shell commands

# FIFO shell example

```
% mkfifo my_fifo
% ls -l my_fifo
prw-------   1 brewsteb user 0 Jul 13 10:48 my_fifo
```

- Since they are files, you can apply most of the common shell commands
  - Ex: read, sort, wc, cut, awk, etc.
- As well as all the common file input/output system calls: open(), read(), write(), etc.

# Opening a FIFO

- There must be at least one reader and one writer process before you can open a FIFO

- Example
  1. Process A calls open(…, O_RDONLY)

     Process A blocks
  2. Process B calls open(…, O_WRONLY)

     Process A and process B resume execution

# FIFOs in C

```
char fifoName[] = "my_fifo";
int fd, newfifo;

newfifo = mkfifo(fifoName, 0644);

fd = open(newfifo, O_RDONLY);
if (fd == -1)
{
   perror("open");
   exit(1);
}
printf("open succeeded");
while ((r = read(fd, buf, sizeof(buf) - 1)) > 0)
{
   buf[r] = '\0';
   printf("%s", buf);
}
```

Blocks until also opened for writing

# Why you might use FIFOs

- Want to build a client-server architecture on a single machine, but you don't want to deal with the complexities of sockets

- You want to access one end of the FIFO with a non-network aware program