Week 08 2

CSC209 Fall 2023

Dr. Demetres (dee) Kostas

November 2, 2023

Announcements

- midterm marks and distribution
  - have been released
  - please have a look
- The drop deadline is Nov. 6th
  - A1 might be back
    - midterm distribution is
      - a better judge
      - perhaps with a higher avg.

Remarks

- if you believe there was an error
  - there are always a few
- email the course address
  - include Utorid and student #
  - explain the error
- you can also come talk to me now
  - or during office hours

Pipes

- a method of inter-process
  - communication
    - beyond return codes...
- Uni-directional buffers
  - bytes go in one end
  - bytes available on other side
    - reading while empty: *waits*
    - writing while full: *waits*

We use the system call `pipe`

- and we manage *file descriptors*
  - for the writing (in) end
  - and reading (out) end
  - *given to us by the system call*
- more primitive system calls
  - `read` and `write`
  - are used with raw file descriptors

Let's try writing some code

`pipe.pdf`

Why pipes versus other IPC?

- largely because they underlie
  - the redirection of byte streams
- one thing pipes *excel* at
  - is interconnection of stream buffers
  - to build more complex flows
    - *assignment 3 in a nutshell*

- consider, how is it that the shell
  - can do something like this:
  - `ls -l | grep keyword`

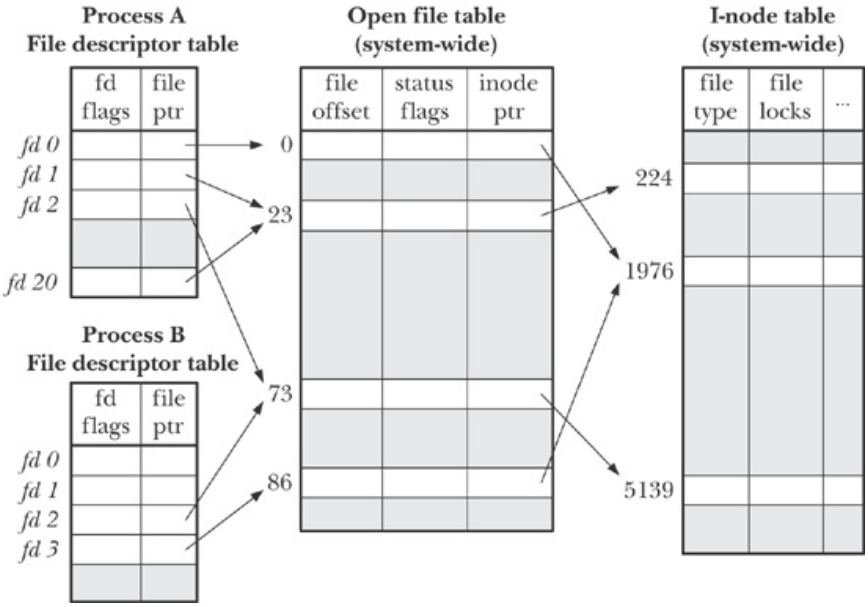
We can ask this more generally...

- you'd like to leverage existing
  - programs into a multi-process app
- can you do this?
  - e.g. use `ls` in your app?
- You could:
  - `fork()`
  - `exec("ls", ...)` in the child
  - but... then what?
    - check the status code?

Duplicating file descriptors (`dup2`)

- you could, set stdout for the `ls` process
  - to be the **write end for a pipe**
  - with a read end that the
    - original parent can access
- you have to parse all the output
  - but this gives us what we want

First, file descriptors again



- Kerrisk 2010 – Figure 5-2

What does `int dup(int oldfd, int newfd);` do?

- it doesn't change the numbers
  - the fd's
- it changes the pointer
  - to point to another file
  - **or** system-level abstraction
    - i.e. a pipe!

What does `int dup(int oldfd, int newfd);` do?

- `newfd` now points to
  - the same thing that `oldfd` does
  - *a new fd for the existing old*
- additionally, the return value

  - will be the `newfd` number
  - or `-1` on error

Our example as snippet

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

if (fork() > 0){ // parent
    close(pipes[1]); // write end
    char buffer[100];
    read(pipes[0], buffer, 100);
    printf("%s", buffer);
    close(pipes[0]);
} else { // child
    close(pipes[0]);
    dup2(pipes[1], STDOUT_FILENO);
    execlp("ls", "ls", "-l", NULL);
}
```

Closing unused fds

- you should do this
  - as soon as you know you don't
    - need an fd
- basically, if not
  - the resources for this object
  - are never free'd!
- in this sense it is memory leak!

Let's write some code

`dup2.pdf`