Subprocess

2024 Fall ECE 344: Operating Systems
Jon Eyolfson

Lecture 8
2.0.0

We Want to Send and Receive Data From a Process

1. Create a new process that launches the command line argument
2. Send the string `Testing\n` to that process
3. Receive any data it writes to standard output

A More Convenient API – `execlp`

```
int execlp(const char *file, const char *arg /* ..., (char *) NULL */);
```

Does not return on success, and -1 on failure (and sets `errno`)

`execlp` will let you skip using string arrays (using C varargs),
and it will also search for executables using the `PATH` environment variable

Our Final APIs — `dup` and `dup2`

```
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

Returns a new file descriptor on success, and `-1` on failure (and sets `errno`)

Copies the file descriptor so `oldfd` and `newfd` refer to the same thing

For `dup` it'll return the lowest file descriptor

For `dup2` it'll atomically close the `newfd` argument (if open),
and then make `newfd` refer to the same thing

Coding Example

Done live!

You can find the template in `08-subprocess` in the `materials` repository

To compile it, run the following commands:

```
cd lectures/08-subprocess # if not already there
meson setup build
meson compile -C build
```

Run the program using: `build/subprocess <program>`

Running with `cat` May Cause Problems

Run the program with the following arguments:

```
build/subprocess uname
build/subprocess cat
```

You have to be careful with the file descriptors!

Why might `cat` not exit when using pipes?