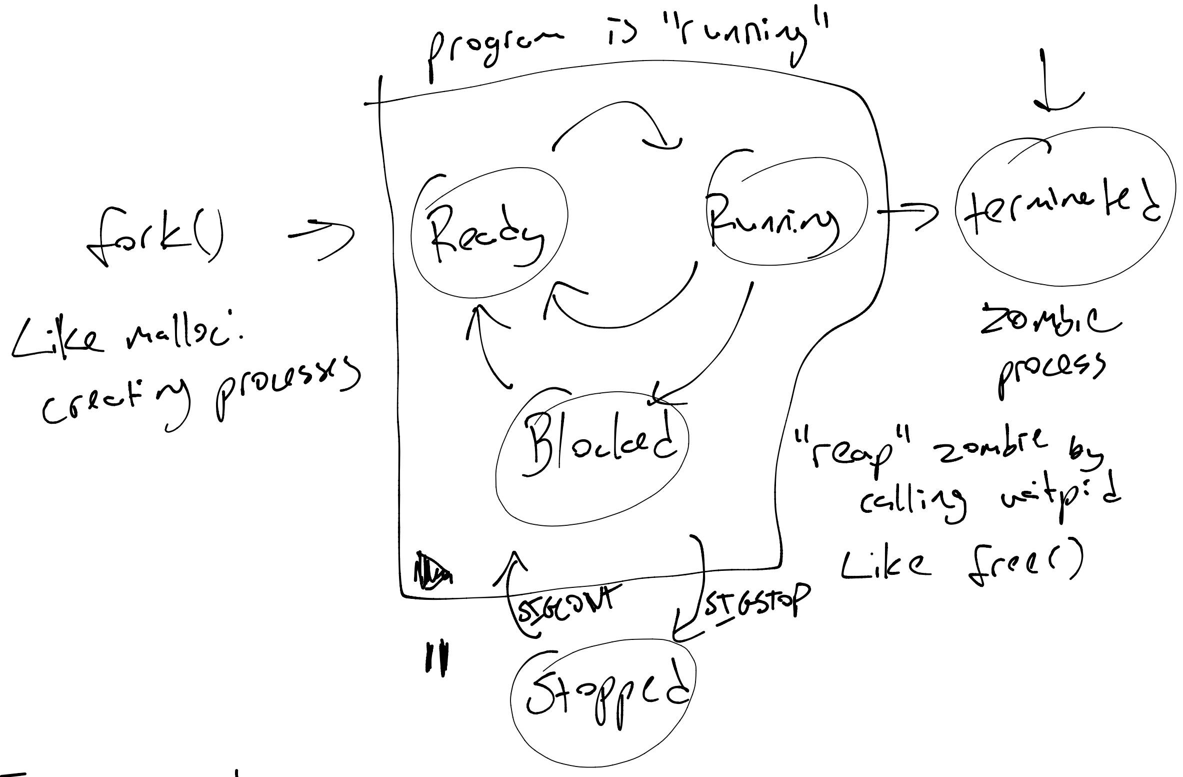


Like malloc: creating processes

zombie process

"reap" zombie by calling waitpid Like free()

Job control

- Send SIGSTOP signal to pause a process
- Send SIGCONT signal to resume a process

On command line:

```
kill -SIGSTOP pid
     -(SIG)CONT
```

Programmatically:

```
kill(pid, SIGSTOP/CONT);
```

waitpid(pid_t pid, int *status, int flags);

child you want to get info about, or -1 if you don't care to be specific

pointer to a collection of bits that tells you info about the child

0: default: wait for child to terminate

WUNTRACED: wait for child to stop

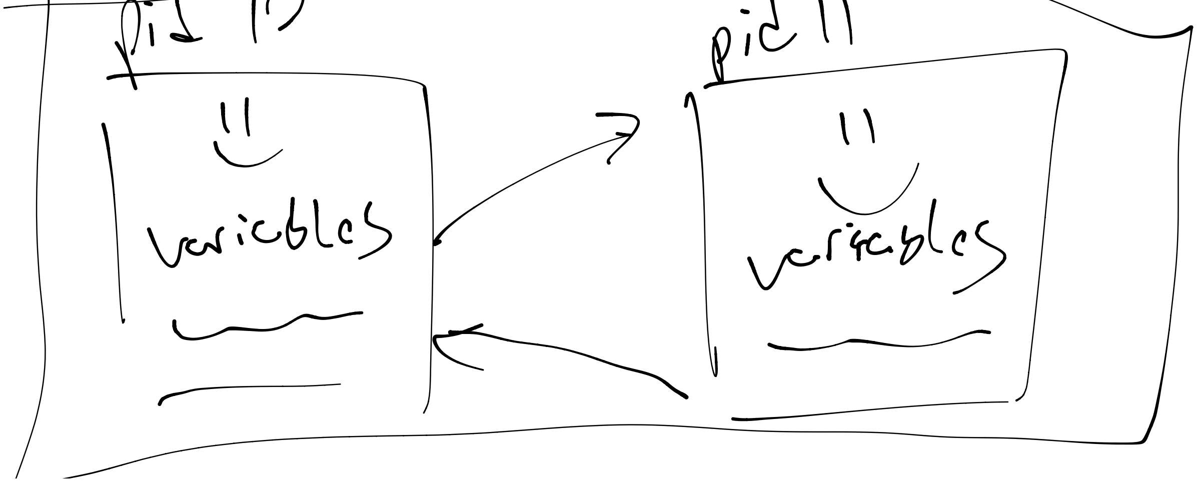
WCONTINUED: wait for child to continue

Execvp:

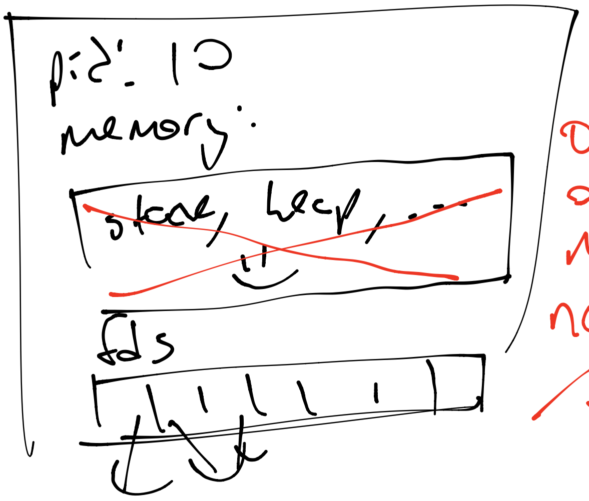
runs an executable on the machine

```
int execvp(const char *executable,
      char * const argv[]);
```

what we might want: → {"make", "clean", NULL}



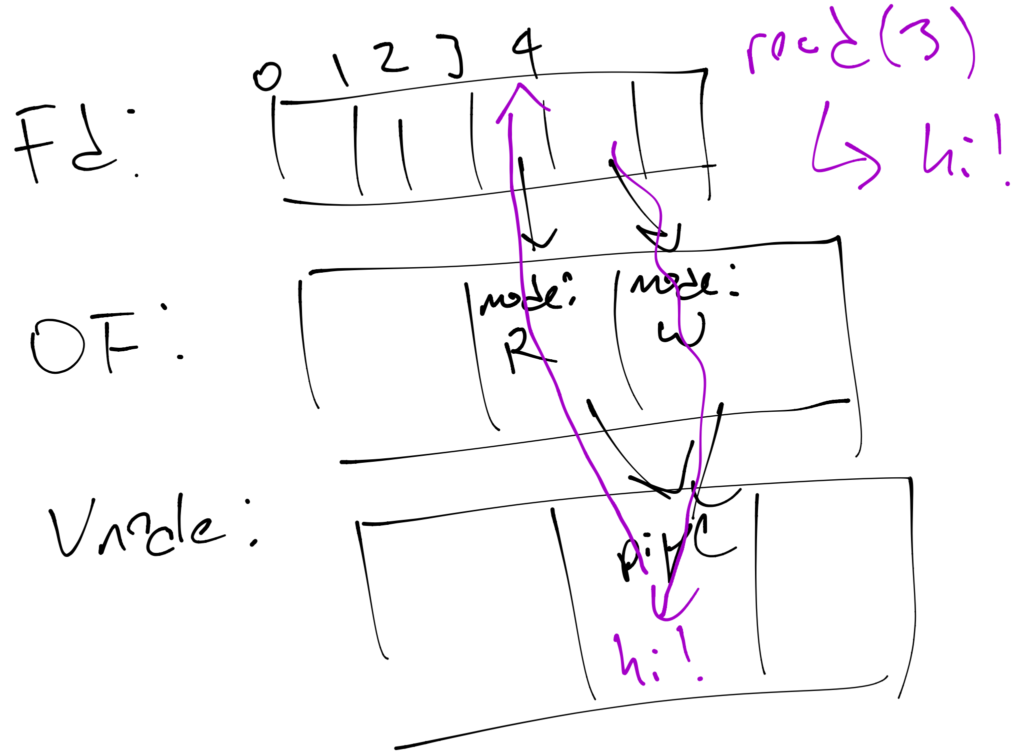
what actually happens



OS throws out virt mem, installs new heap/stack/text segment

Pipes

Special file in memory with a write end that you can write to, and a read end that echoes those bytes

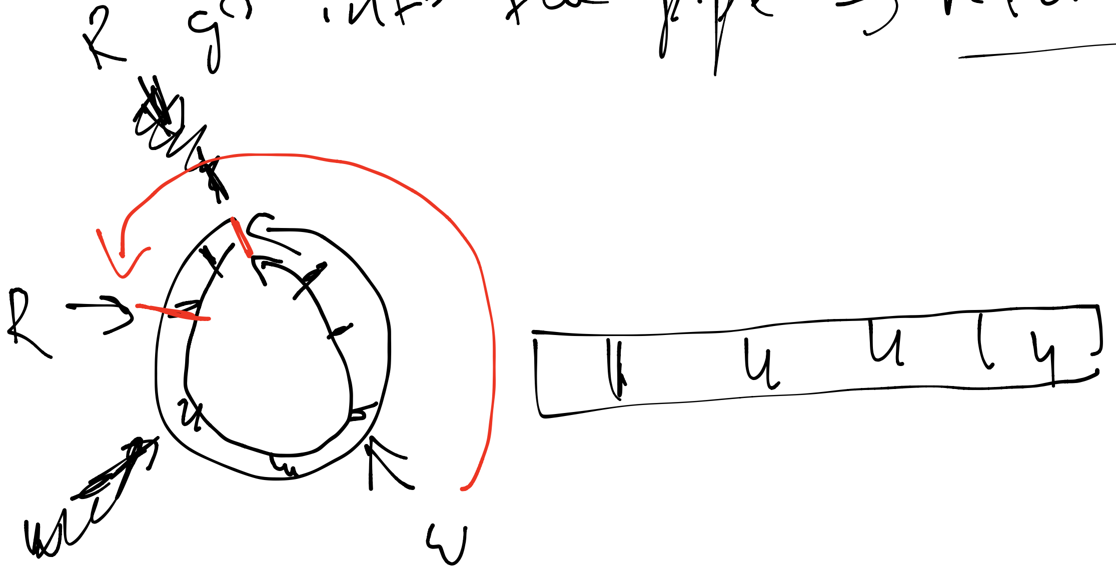


write is right

learn to read before you learn to write

read:

- If nothing is in the pipe, waits for at least 1 byte to be available
- If all write file descriptors are closed, no more data can ever go into the pipe ⇒ returns 0


R
R
W
W
4
4
4
4
4
4