# CS 3650 – Computer Systems
Spring 2024

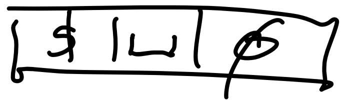
Peter Desnoyers

Lecture 7, Tue Jan 30 2024

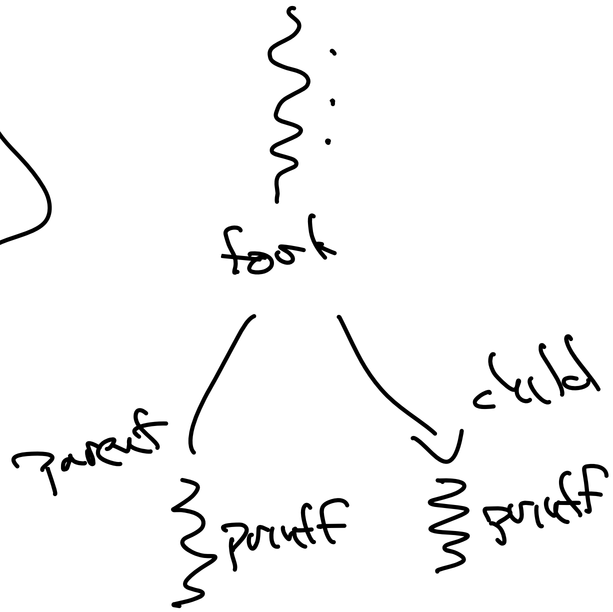
# The simplest shell

```
while 1
    write(1, "$ ", 2)
    read line, sizeof(line) from fd ∅
    -> parse line => command, args
    int pid = fork();
    if pid == 0
        execvp(command, args)
           -> exit
    else
        waitpid(pid, &status)

    ~~~~ (parent)
```

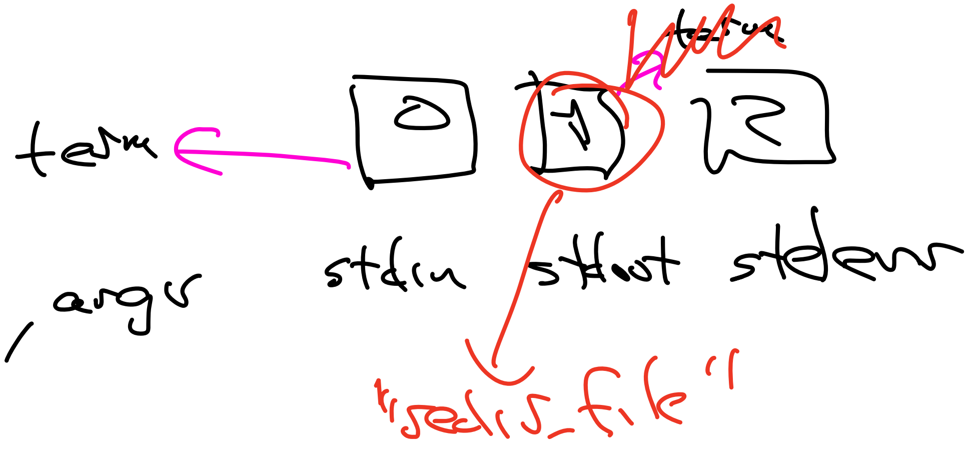


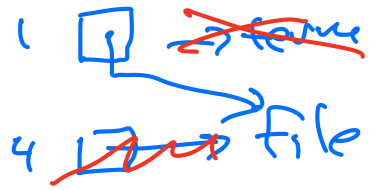
fgets



```
while 1
  write(1, "$ ", 2)
   readcommand -> cmd, argv
   int pid = fork()
   if pid == 0
      close(1)
      open(redir_file, O_CREAT | TRUNC | WRONLY, 0666)
      execvp(cmd, argv)  ------------------>  main() {
                                                printf("hello
                                                   world\n")
                                              }

int fd = open(...)
dup2(fd, 1)   <-  close 1
                  copy fd into fd 1
                  close fd
close(fd)
```





## Factoring stuff in your shell

```
launch (cmd, argv, <i/o redirection>)
                                            in? in_fd
                                            out? out_fd
    fork; child: {
        if redir input :
            dup2(in_fd, 0)
            close (in_fd)

        if out?
            dup2 (out_fd, 1)
            close (out_fd)
```

```
launch (cmd, argv, char * infile,
                   char * outfile)
NULL : don't redirect

  if pid == 0 :
     if infile ≠ NULL
        open / dup / close
     if outfile ≠ NULL
        open / dup / close

     execvp( . . . )
```

f( . . . )
fork
fork
f( . . . )

internal commands :
→ no need for transparent redirection

## redirect strategies:

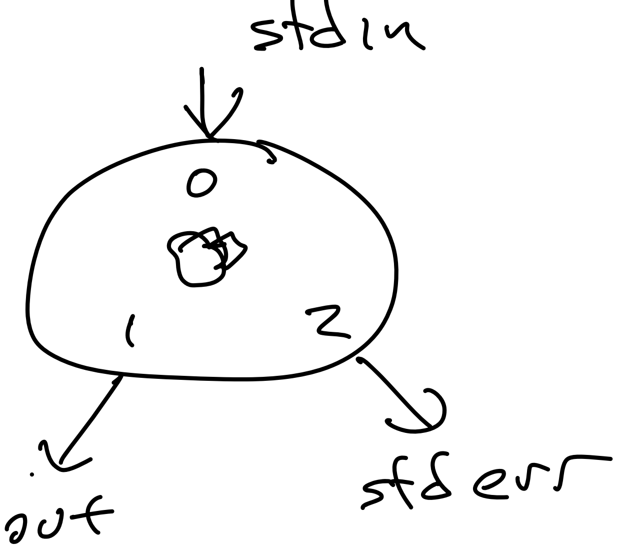
(eg NULL = no redir)

$$\begin{tikzcd}\text{} \arrow[r, "\text{char * filename}"] & \cdots \text{open @ last minute}\end{tikzcd}$$

open

$$\begin{tikzcd}\text{} \arrow[r, "\text{int fd}"] & \text{dup, etc.}\end{tikzcd}$$

---

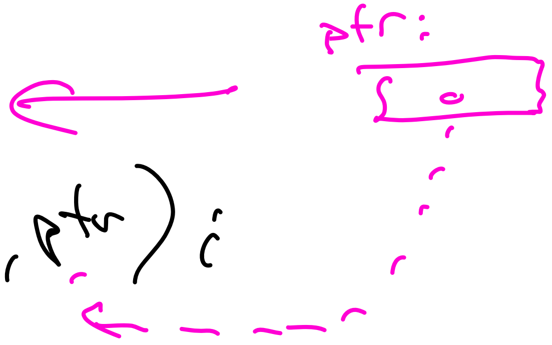


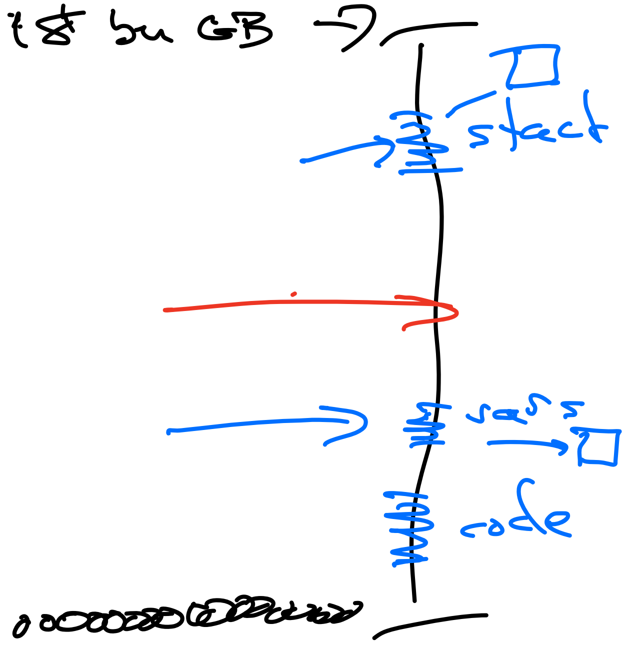
# segfaults & uninitialized variables

## what is it?

## how does it happen?

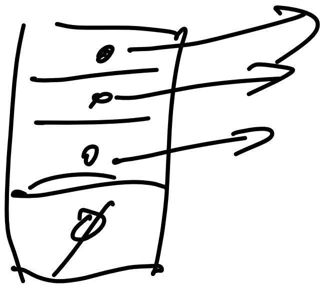
```
foo() {
    char *ptr;
    printf("%s\n", ptr);
```

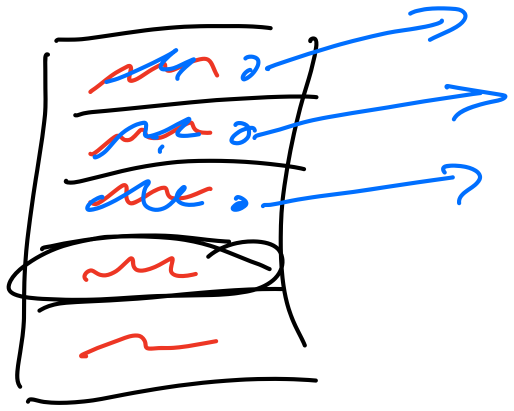




char *ptr=NULL;

char argv[10];





# Debugger

```
gdb sh3650
    :
gdb) run
       :
```

| | |
|---|---|
| n | next |
| s | step |
| p | print |
| bt | backtrace |
| up | |
| down | |
| l | list |
| l | <line#> |
| b | <line #> |
| | function |
| | file : line# |

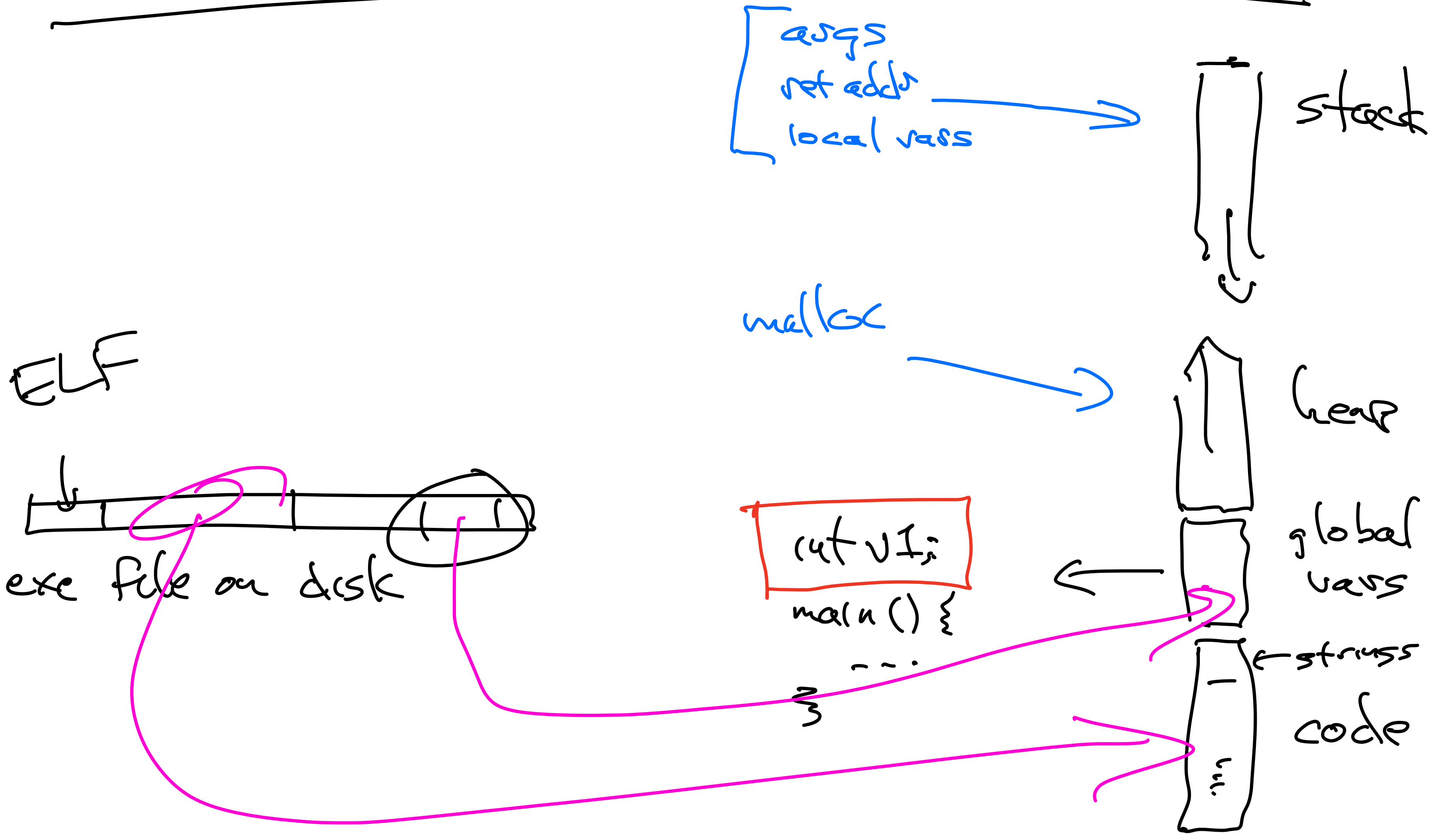
info break

delete <n> ← break#

```
strace -f ./sh3650 file.cmds
```

^ "Follow"

# Address space & virt. memory



args
ret addr
local vars
stack
malloc
heap
ELF
exe file on disk
int v1;
main() {
...
}
global vars
strings
code