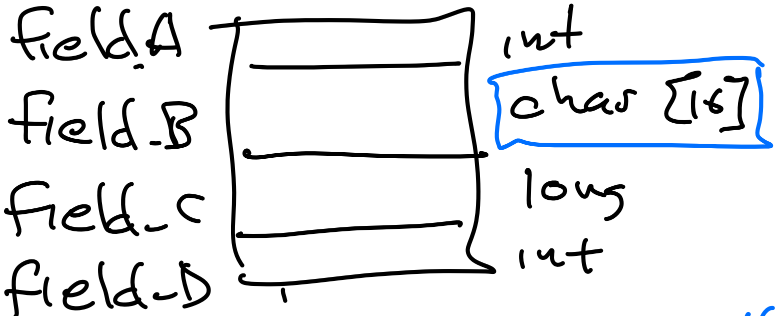
CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 5, Tue Jan 23 2024

File and process system calls

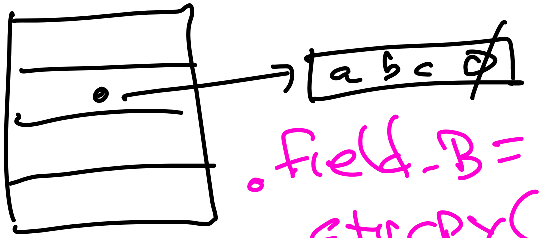
C language stuff

struct - a minimal object



No malloc

```
struct abc var;
   var.field_A = 10;
   strcpy(var.field_B, "abc")
```



.field_B = malloc(4)
strcpy(.. field_B, "abc")

```
struct abc {
    int field_A;
    char field_B[16];
    long field_C;
    int field_D;
};
```

```
struct abc *ptr =
    malloc(sizeof(*ptr))

ptr->field_A = 10

strcpy(ptr->field_B,
        "abc")
```

sizeof(type)

sizeof (expression) ≡ sizeof(typeof(expr))

type *ptr = malloc(sizeof(*ptr))

~~malloc(sizeof(ptr))~~

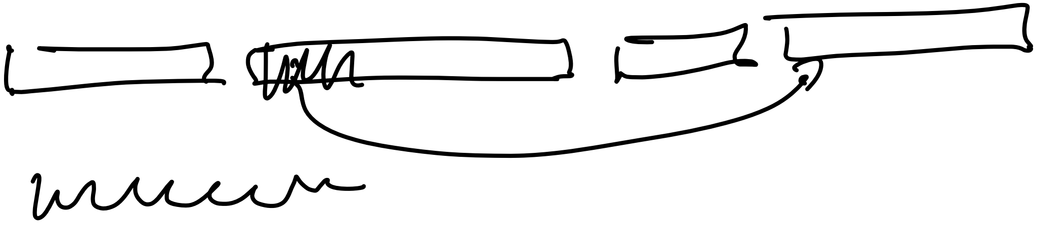
malloc(sizeof(struct abc))

0

'\0'

'0'

"malloc: free list corrupted"



int bytes = sizeof(struct abc)

= sizeof(*ptr) . .

~~= 32~~

ptr = malloc(bytes)

malloc(int size)

32

ptr = malloc($\underbrace{\text{sizeof}(\dots)}_{\text{integer expression}}$)

expressions: ints 1, 2, 3 . . . (decimal)

0xA10, 0xA11, . . .

010 = 8 011 = 9 . . .

single quote: 'A' '\n' ← newline

operators: + - * / % & | ^ (and, or, exclusive or) operators : + - * / % & | ^ (and, or, exclusive or)

← bitwise logical ops

binary :

```
0 1 1 0 1 1 0
1 0 1 1 0 0 1

0 0 1 0 0 0 0    &
1 1 1 1 1 1 1    |
1 1 0 1 1 1 1    ^
```

boolean ops:

bool a = true, b = false

a || b (true)

a && b (false)

a == 0

int

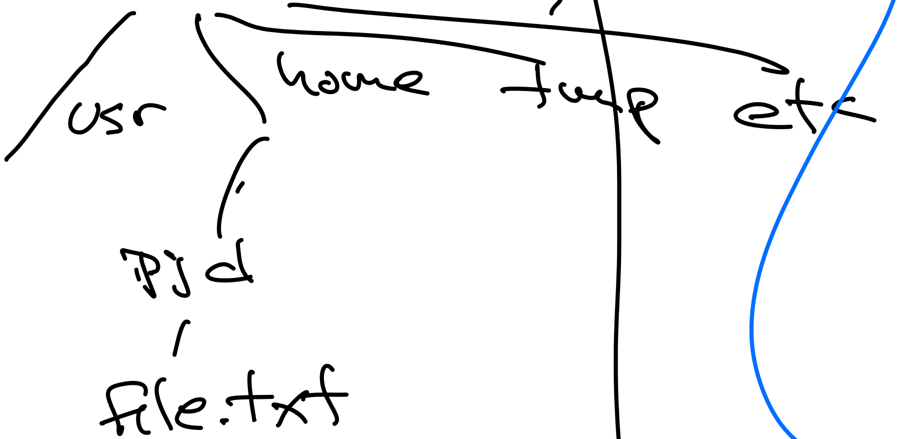
true : non-zero

false : zero

System calls

open / close / read / write / lseek

```
open( char * path,      ← e.g. "file.txt" ← local
      int flags
    { int permission )         "/home/pjd/file.txt"
                                ↗
                            absolute
```



0777

```
O_RDONLY
O_WRONLY |
  O_CREAT |
  O_TRUNC
```

close

read (fd, void *buf, int len)

←

bytes read | -1

write (fd, buf, len)

←

len | -1

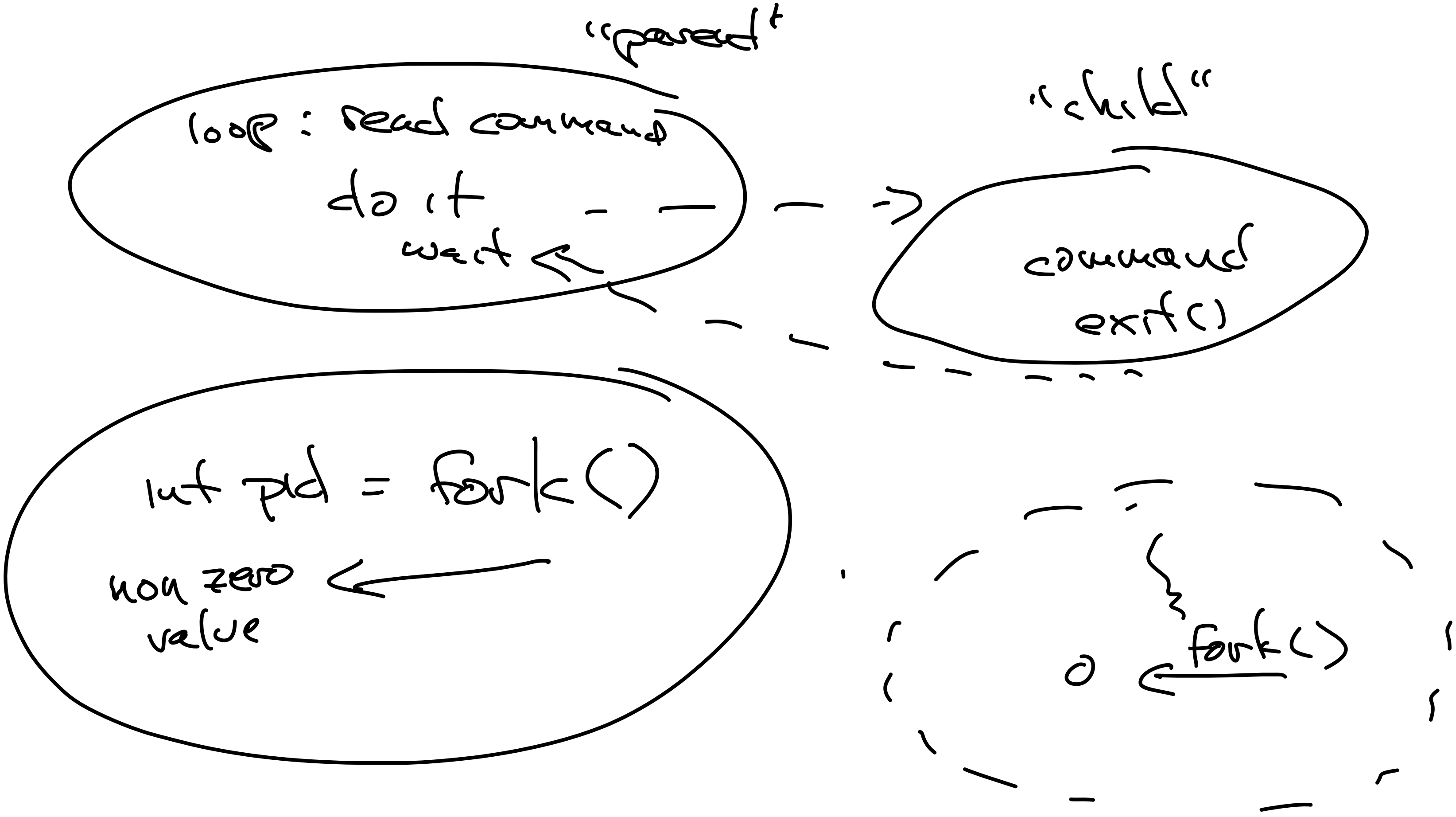
```
int fd = open("file",
              O_RDONLY)

char buf[1024]

int len = read( fd,
                1024)

fd2 = open("file2",
           W RONLY | CREAT |
           TRUNC, 0777)

loop :
    len = read
int err = write( . .  )
```

Processes

ps -aux

process status

process = running program



```
int pid = fork()
if (pid == 0) {
      exec("file", arguments)
}
else {
      wait(pid)
}
```

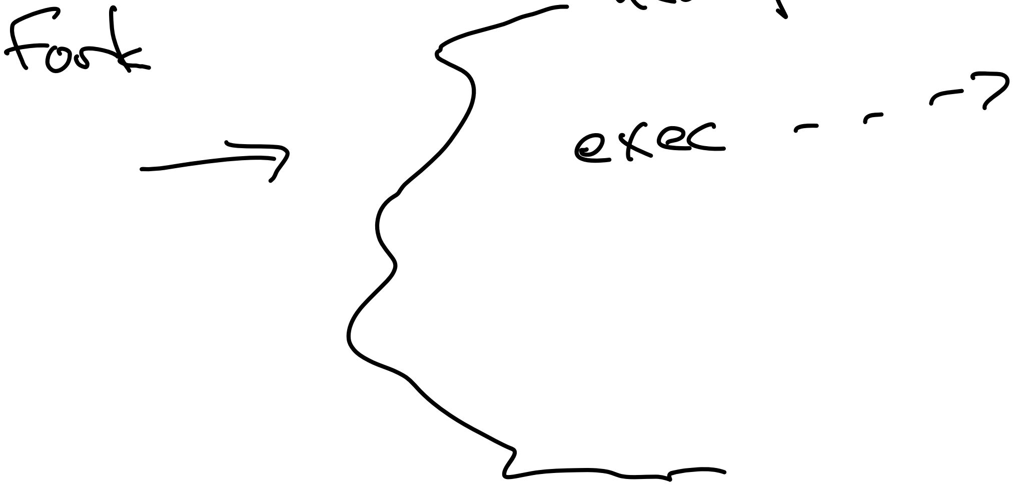
child →

parent →

i.e. argc, argv

("ls", ...)

"/usr/bin/ls"



new program

int val = main(...);
...

exit(val)

startup library code

I/O redirection

```
$ ls
  ⋮
$ ls > file.out
$ cat file.out
$ grep zip < file.out
```

command

redirect stdin

```
$ grep zip
  ...
  ^D ← end of file
```

```
$ ls /usr/bin | grep zip
```

"pipe"

ls : list

ls -l : list long fmt

↖ lower case L

cat : type?

grep : search

less ↖ pun on "more"

q to quit