CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

Lecture 27, Tue Apr 16 2024

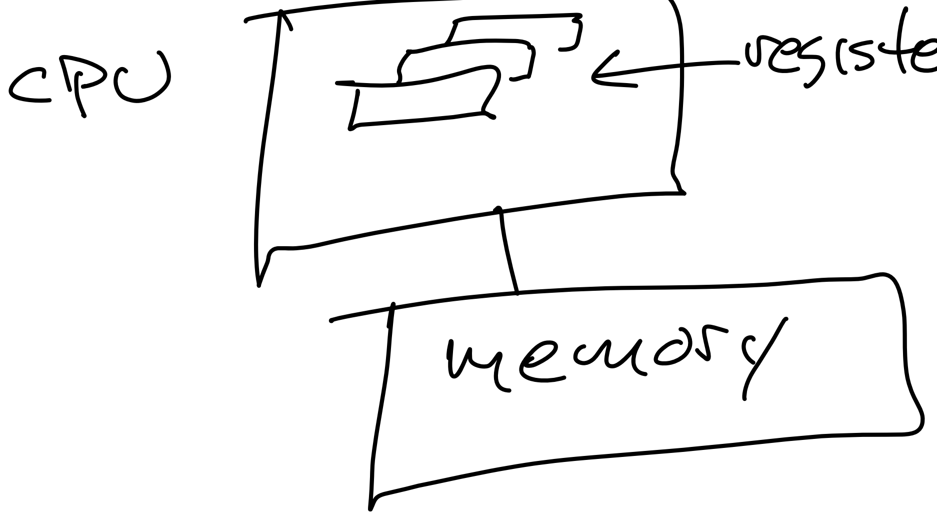
Final exam review

- computer system basics
- C and bugs

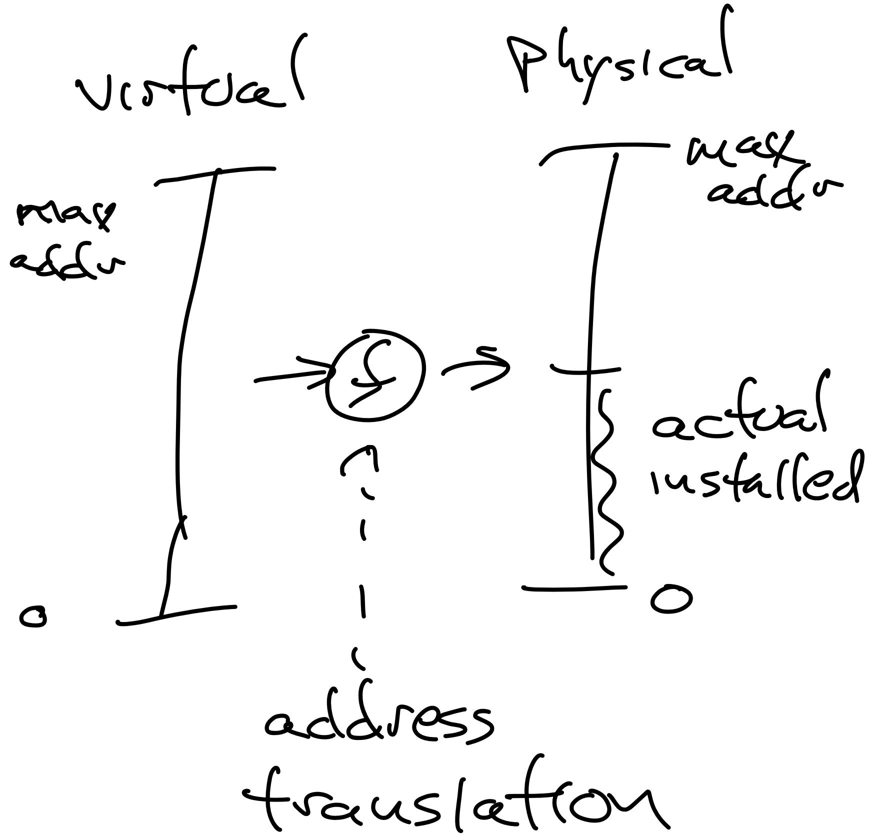
} a bit of mutex, concurrency, shell, fork, etc.

- authentication & access control
- file systems
- lab 4
- lab 5

Computer system basics



e.g. - executes instructions



max addr:

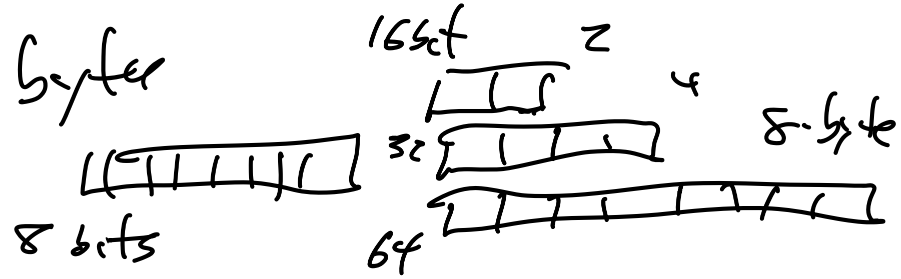
32 bits: $2^{32}-1 = \text{FFFFFFFF}$

~~2~~ = 4GB

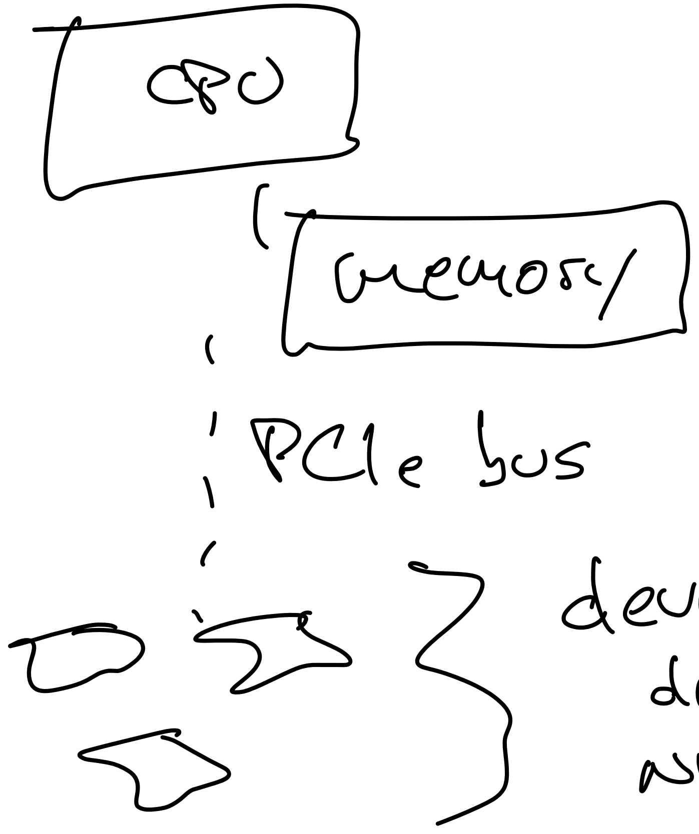
64 bits: $2^{64}-1$ = lots of Fs

= lots of TB

(16 PB??)



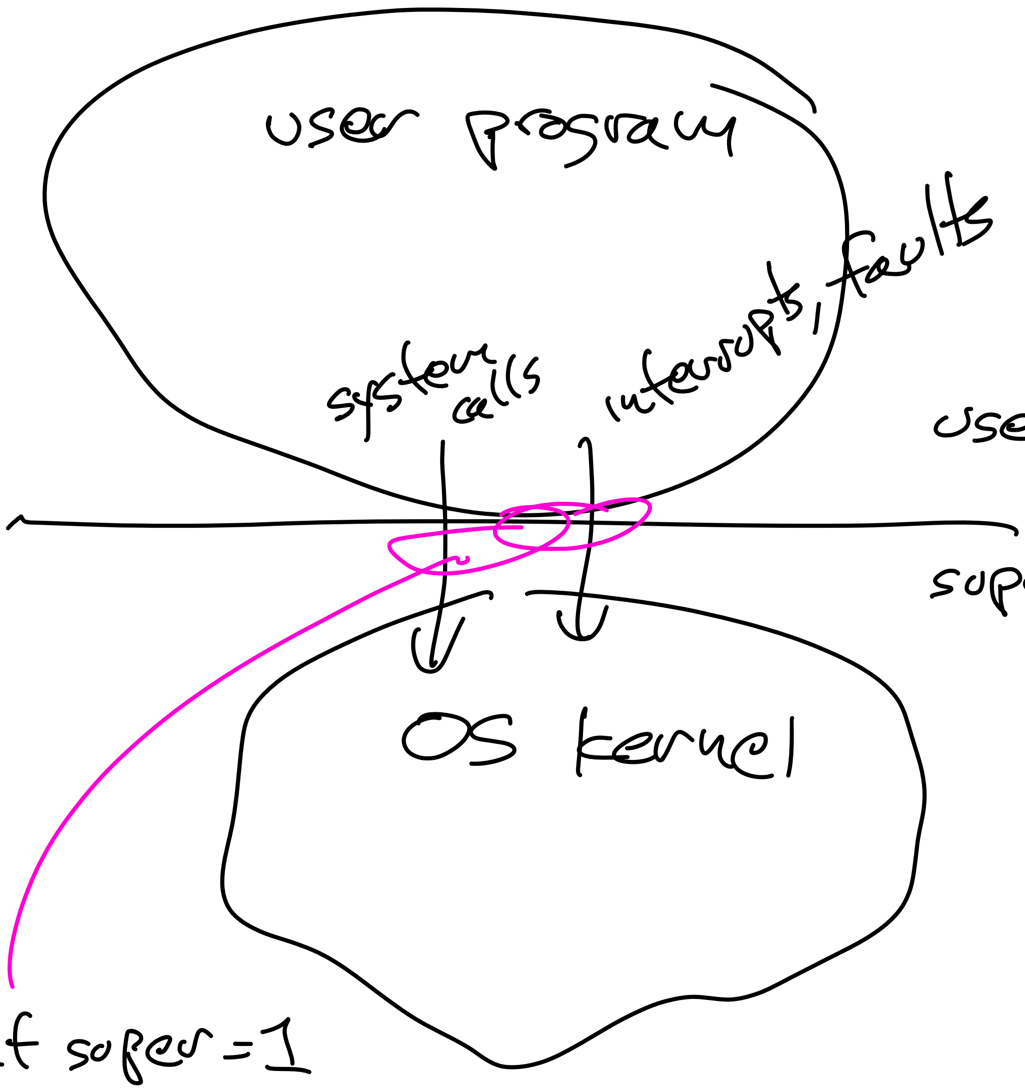
Input/output



devices:

- disk controller
- nvme drive
- network adapter
- graphics card
- USB controller

software picture



user mode

supervisor mode

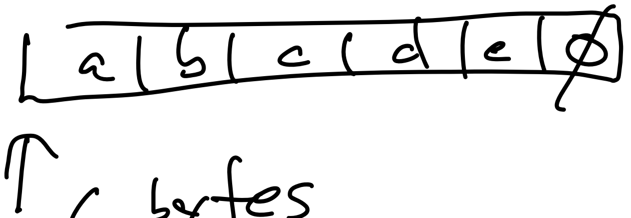
1) set super = 1
2) jump to trusted handler

user mode:
can't subvert protection

supervisor mode:
can configure protection
(eg context switch)

C programming

strings



null-terminated strings

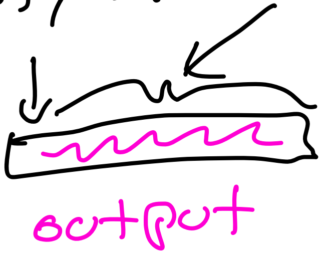
6 bytes

strlen() = 5

C caller-allocates pattern

char buf[128] (or malloc)

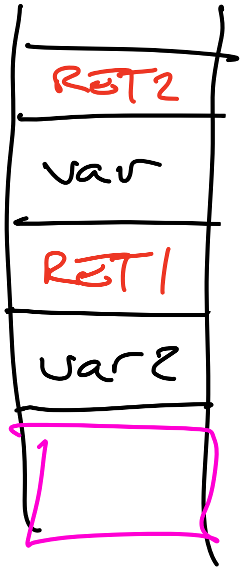
val = fgets(buf, sizeof(buf), stdin)



stack vs heap

```
f() {                 g() {
   int var;              int var2;
   g();                  ...
}  printf()           }
```

RET 1



local variable scope:
from definition to end of block/function

```
char *f() {
   char buf[16]
   return buf;
}
```

very bad

Mutexes and condition variables

mutex protects data

```
struct x {
    mutex m;
    <stuff>
}

modify-x (struct x *p) {
    lock (~~&~~p->m)
    ~ do it ~
    unlock(p->m)
}
```

```
access_x(..) {
  lock
  tmp = ...
  unlock
  return tmp
}
```

single thread correctness pattern

```
if invariant(object):
    modify(object)
```

→ invariant still true

threads lock mutexes
then release them

→ deadlock if they loop

classic deadlock

```
~           ~
lock A      lock B
lock B      lock A
~
unlockB     unlockA
unlockA     unlockB
```

Unix Access control

process has:
- user id
- {group ids}

file has:
- owner (user id)
- group (group id)

owner perms	group perms	other ("world"
R W X	R W X	R W X

```
if proc.uid = file.owner
    check owner perm
else if file.group in proc.groups:
    check group perm
else
    check other
```

Octal encoding:

$$\begin{tikzcd}
 & \text{owner} \arrow[dl] & \text{group} \arrow[dl] & \text{world} \arrow[dl] \\
7 & 5 & 4 &
\end{tikzcd}$$

7 = RWX

5 = R-X

4 = R--

111	= 7
110	= 6
101	= 5
100	= 4
011	= 3
010	= 2
001	= 1
000	= 0

700 = owner RWX
group ---
world ---

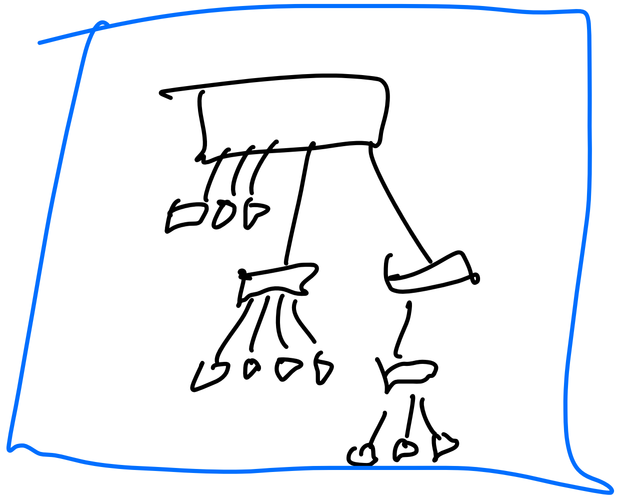
File systems

files, directories

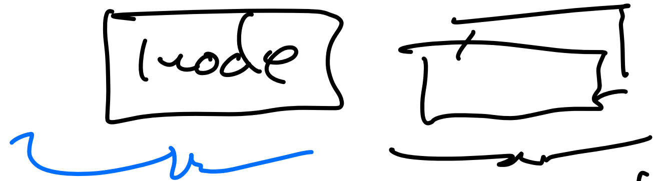
→ CD ROM
→ MSDOS (FAT)
→ unix

lab4 = minimal unix

not on exam?



unix: file/dir =



meta data:
owner
perms
size
block ptrs
⋮

data blocks

user
read/write
offset, len (byte)

{ file sys

blocks (eg 4K)

disk

Lab 4 question

given block 1 inode . . . . .

block 2

block 3 . . .

1) draw file system hierarchy

2) for read (path = /a/b, offset = X, len = Y)

what blocks get read?

→ assume nothing in memory

Lab 5 questions

fdset
FD_ZERO
FD_SET
FD_ISSET

} understand how you use them to encode e.g. {fd1, fd2}

{read fd set} → select(x, rfds, 0, 0, 0) → {readable fds}

"readable" means read(fd) won't block

other stuff

fork:

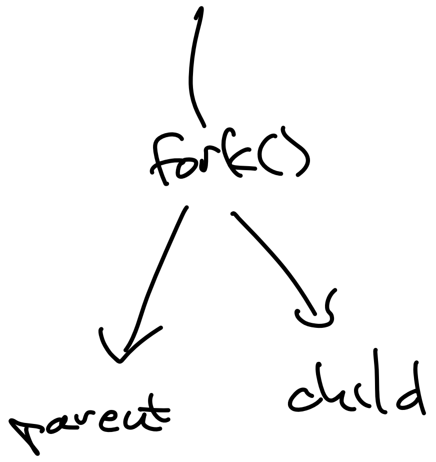
```
        var = fork()
        if (var == 0) {
1)          in child
        }
        else {
2)          in parent
        }
3)      printf("done \n")    ← parent AND child
```

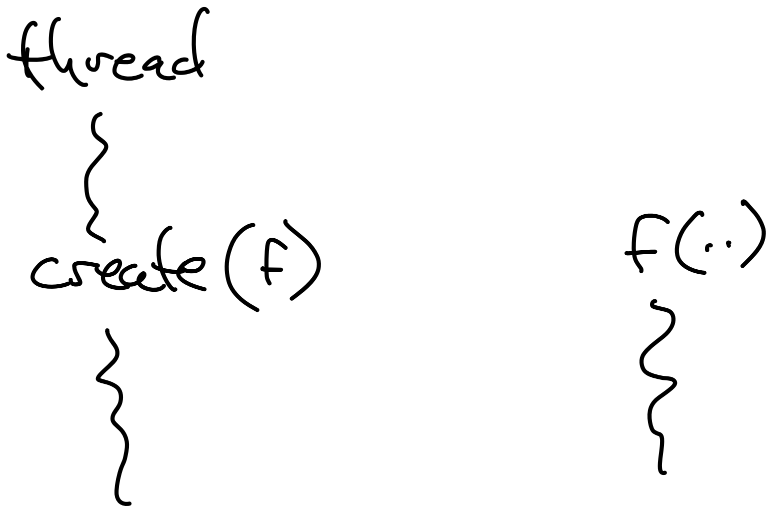
thread:

```
thread_funct (..) {
   ~
}
```

```
pthread_create (thread_func, ...)
```

fork:





Note policy : 1 page
(will check double-sided)

Wednesday 3:30-5:30

WU F 020