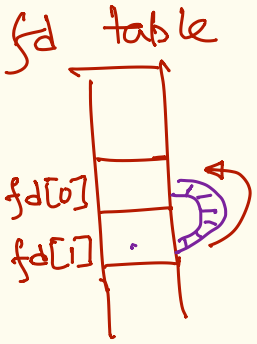
CS744 Lecture 8

~ fork, exec, wait, open, close, read, write, lseek, dup, dup2, pipe, getpid, getppid, waitpid, exit signal, kill ....

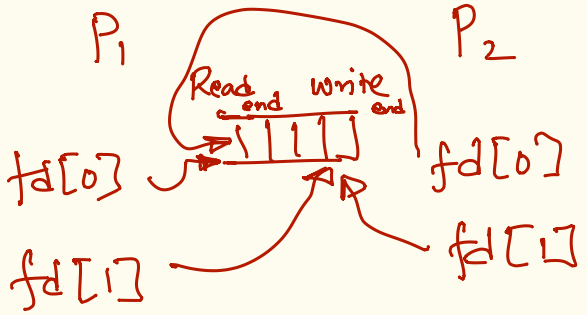
---

pipe

```
int fd[2];
pipe(fd);
fork();
```

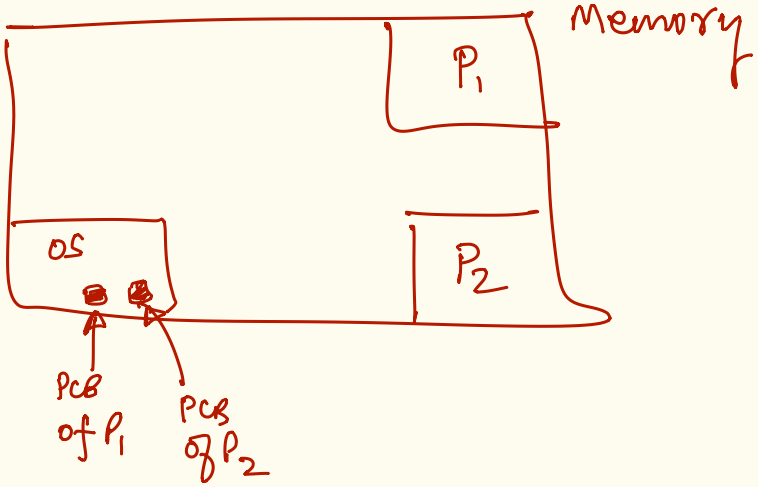


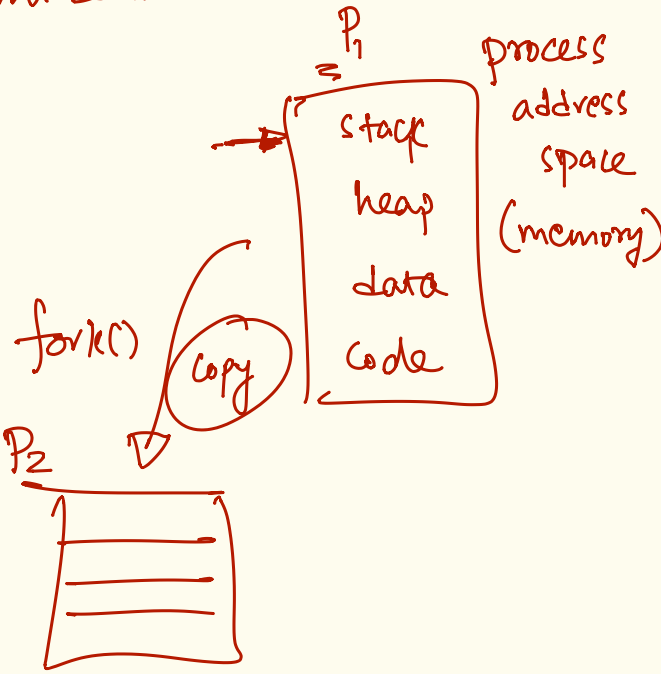
fd[0] ~ read end
fd[1] ~ write end



CoW ~ copy-on-write.

~ fork/exec related optimization.





```
int a[1e7];
fork();
P1                          P2
*                           *a[0]++;
{ a[0]++;                   p=getpid();
wait()                      a[p]++;
}                           exit();
```

mov eax, #mem
inc eax
mov #mem, eax

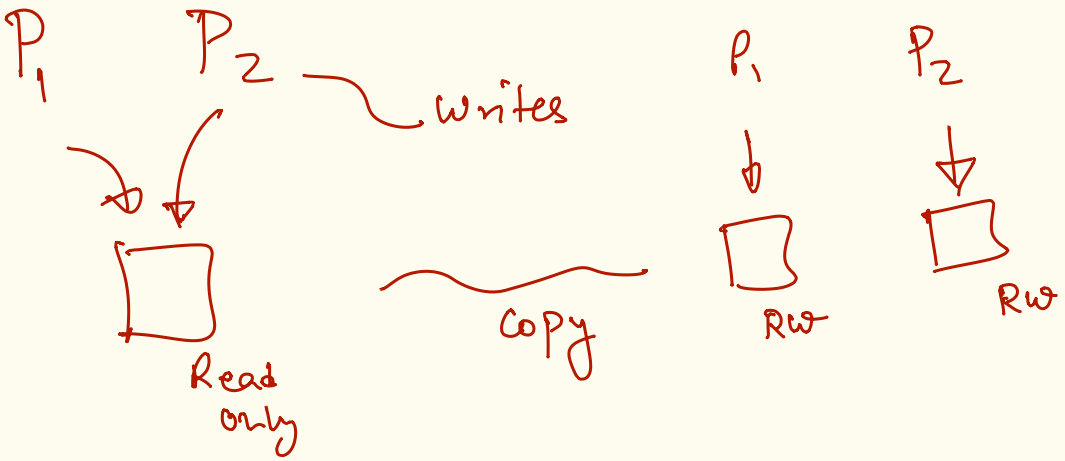
COW

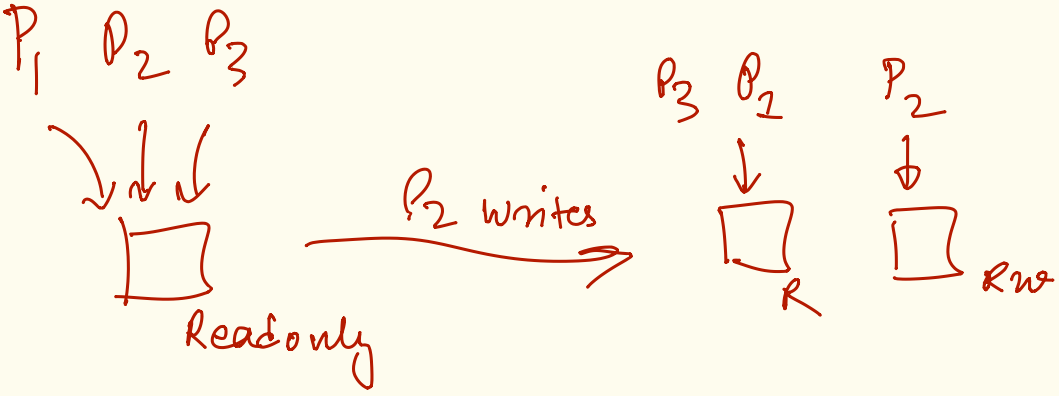
copy-on-write

~ share all memory regions on fork()

~ mark them read-only

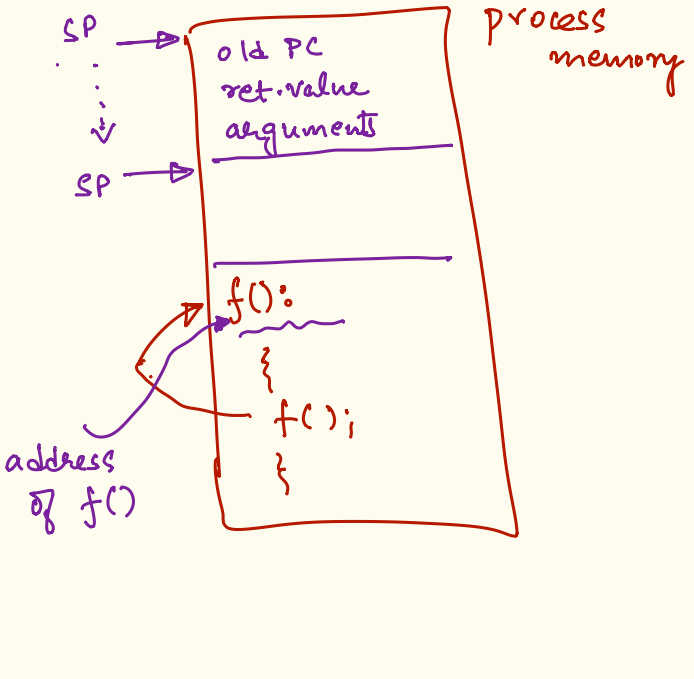
~ on write to read-only region generates an interrupt (page fault)

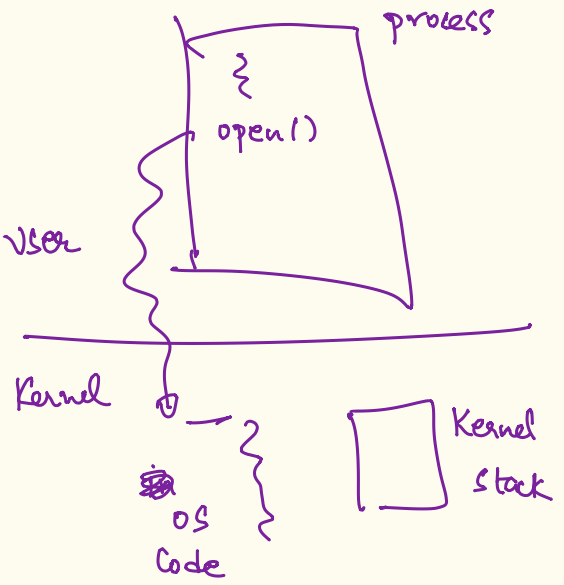




System calls

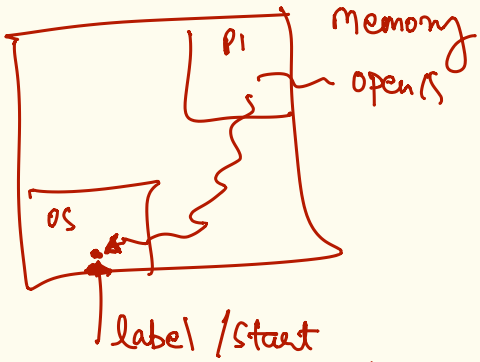
function calls vs system call





design of system calls

- how to invoke a system call?
- how to change privilege levels?
- how to pass arguments & get return values?
- how to identify which system call?
- how to return from system call?
- how to handle context of calling process?



(*) Ans: Short ans: no ISA support ⇒ no system call!

H/w assisted system call invocation:

(1) explicit s/w interrupt via an instruction

int 0x80 —— x86 instruction

opcode ↑ ↑ argument (interrupt vector)

- switch CPU to PL=0
- saves context of user process on kernel stack
- switch to kernel stack
- jmps to an interrupt handler.