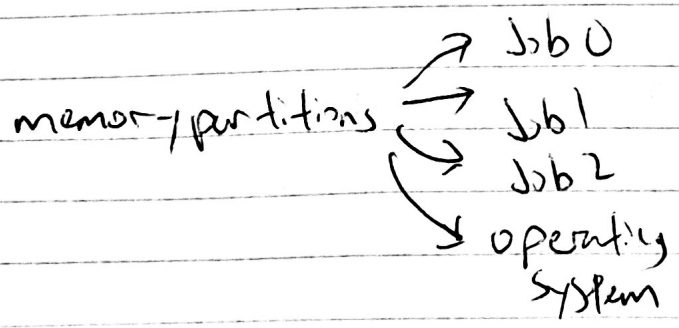
First gen OS : direct input
run 1 job at a time
- enter into comp
- run
- record results

problem : lots of wasted computer time
- computer idle during first & last step

2nd gen : Batch Systems
read cards into input tape
put input tape on 7094
perform computation

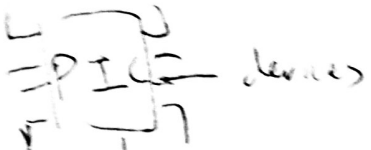
spooling
disks enabled simultaneous peripheral operation online
- computer overlapped I/O of 1 job w/ execution of another
- better utilization of CPU but only 1 job run at a time

3rd gen : multiprogramming

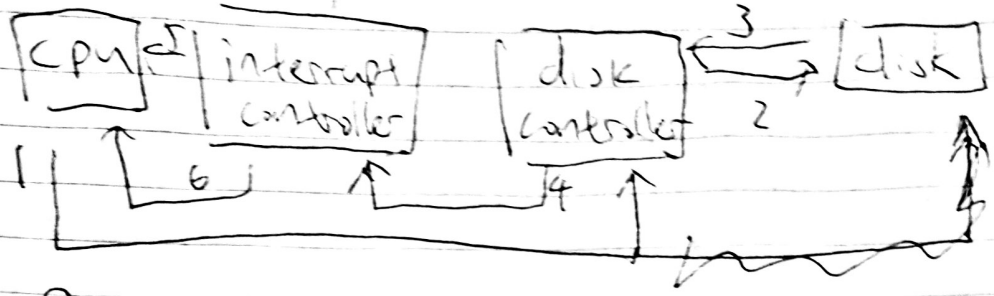


multiple jobs in memory
- protected from each other

resources split between jobs
still not interactive



anatomy of device request



Processes : program in execution
- address space (memory) program can use
- state (registers, PC, stack pointer)

- OS keeps track of all processes

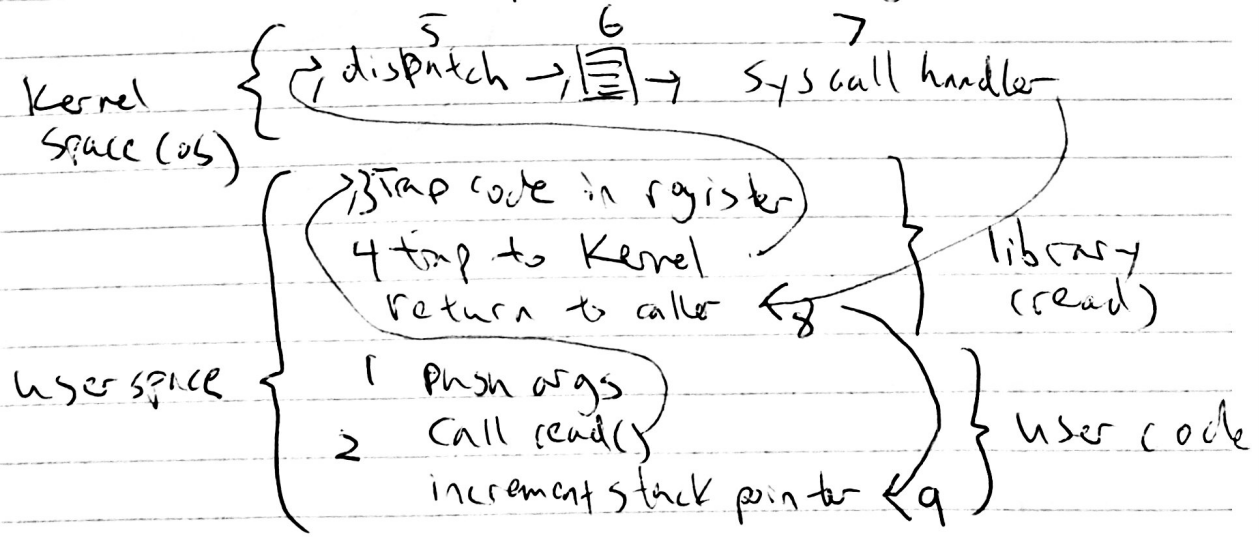
3 segments
- text: program code
- data: program data
  - statically declared vars
  - mem from malloc
- stack
  - automatic variables
  - procedure call info

Deadlock
processes use same resource

Interprocess Communication
processes exchange info
- network
- pipe (A writes to pipe, B reads from pipe)

System calls
- OS runs in privileged mode
  some operations permitted only in system mode
  ex. access device like disk / network card
- user programs can't do these operations
- programs want OS to do service like
  - access file
  - create process
  accomplished by system call

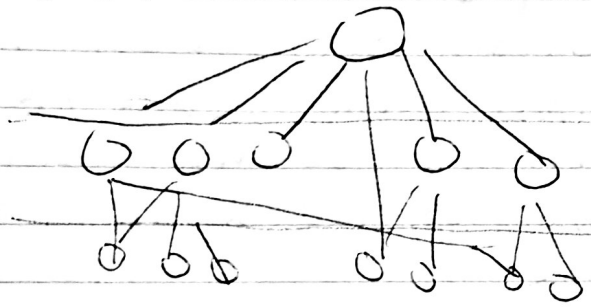
How system calls work
- user program enters supervisor mode
- program passes info to OS
- OS performs service if
  - OS is able
  - service is permitted for program



OS structure

memory management
process management
device drives
file system

Monolithic



all of OS is 1 b.g program
- any piece can access any piece
easy to write, harder to get right

Layered OS