CS 3214 lecture #8 job control shells

Processes: OS abstraction of a running program
(safely multiplex H/W)

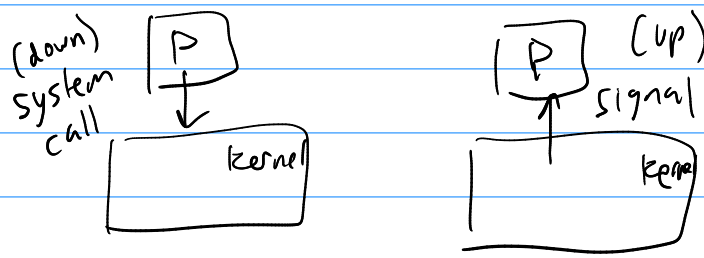
OS provider APIs for

- unified access to resource

  open
  read
  write
  close
  } fd

- creating new processes & running new programs
  fork/exec
- processes to communicate
  pipes

- communicate w/ the kernel



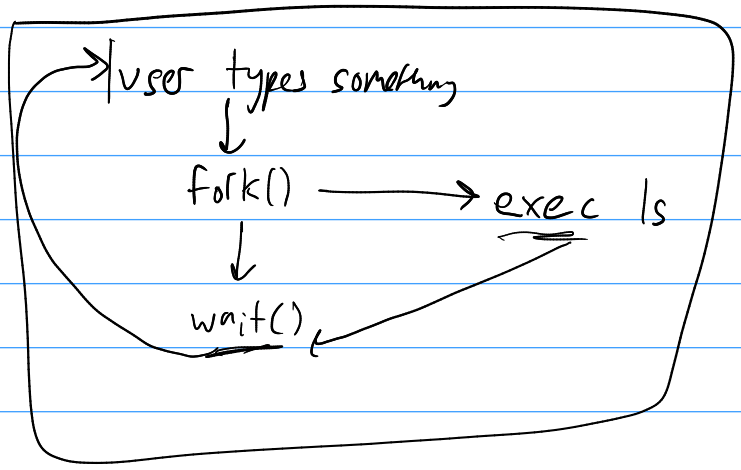
Creation
communication
signalling
} OS providing everything needed for a user program to manage/control lifecycle of all programs & capture user intent

shell

jobs control shell

cat foo | grep ≥ 1 processes in a job

- read/eval loop
  - user input
  - file builtin
- Convenient way to pipe programs together



- fg vs. bg
- user can control job state (^C ^Z)
- shell tells user what happened to job

Users perceive state of jobs

- fg — one job is currently taking input
- bg — executing, but user can still use the shells
- stopped — not fg or bg

OS view:

minimal notion of fg/bg (+)

process groups

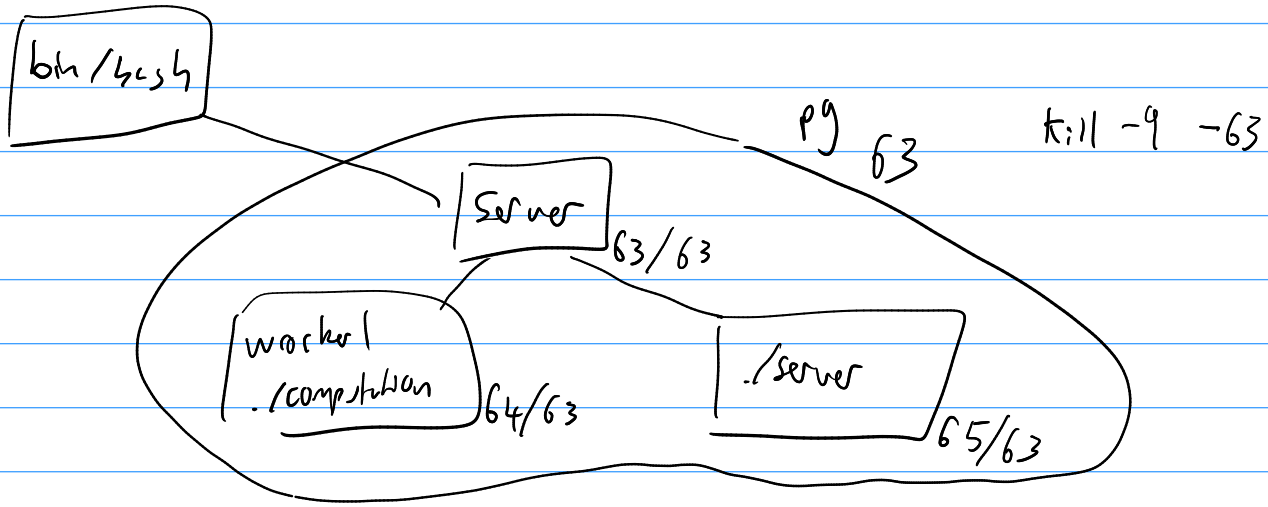
- every process has 1 group
- create a new process group (the leader) & (pgid == pid)
- any process may join or be assigned to a pg

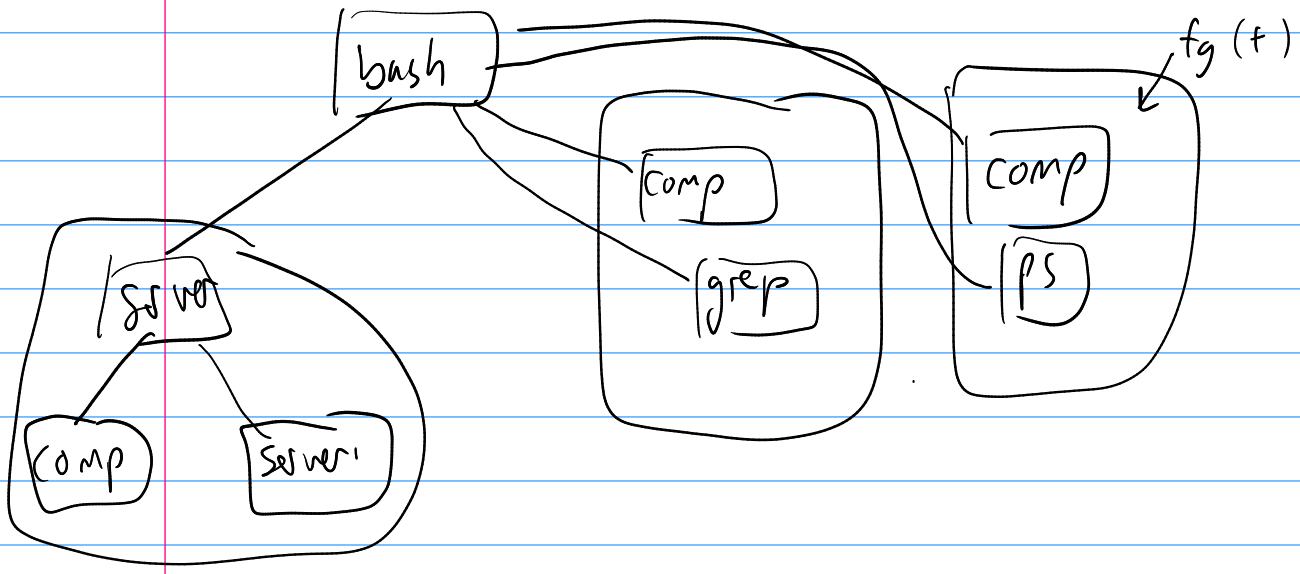
why? signals to entire group

job control shell

- arrange processes into process groups (jobs)
- forked children inherit pg by default

^C SIGTTOU
^Z SIGTTIN





Undefined Behavior (UB) C

spec: SHOULD
SHOULD NOT
} compiler will take advantage of

Result: surprising & horrible bugs

Signed integer overflow

x+1 > x compiler always true!

MAXINT

x * 2 / 2 compiler: always x!

why does C have UB?

essential for some compiler opt.

compiler powers vs programmer responsibility

contract: know UB in your language

in C -Werror -Wall

valgrind Memcheck tool

clang experimental -f-catch-undefined-behavior

-O0