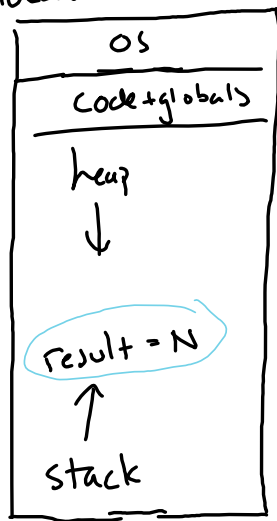


Process #33

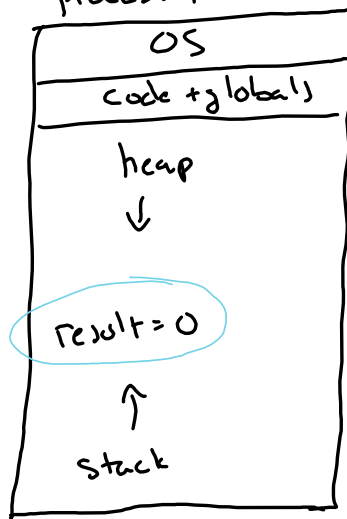


copy entire process  
 $\text{pid\_t result} = \text{fork}();$

only difference  
 is pid and  
 result of fork!

both keep running!

Process #N



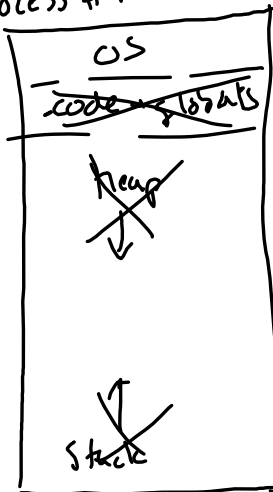
$\text{pid\_t fork}()$

Create a new process that is a copy  
 of the current process.

In the new process, return 0.

In the original "parent" process, return the  
 pid of the child.

Process #N



code + globals for ls

empty heap

$\text{execvp}("ls", \{\text{NULL}\})$

stack at start of  
 main() for ls

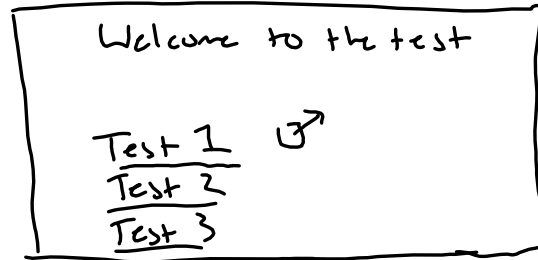
$\text{execvp}(\text{cmd}, \text{args})$

Replace current process with the state  
 of a freshly-started program  
 cmd with args

(args is a NULL-terminated array of  $\text{char}^*$ )

## Reminders

- Make-up exams next week
- You sign up for 1-2 slots

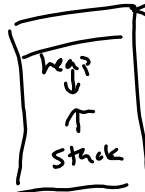


- Review quiz this week
- PA4 due tonight (rough goal → grade by Tue, resubmit Mon)
- PA5 no resubmit, no design Q

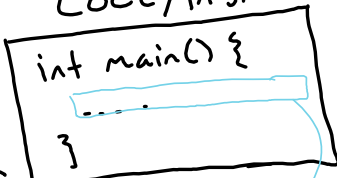
# Processes - What is a process?

What are parts of a process?

Memory



code/instructions



current instruction

system resources

- sockets
- stdin
- stdout

Not part of process!  
gcc before our program starts!

A process is the memory, (current) instructions, and system resources for a running program.

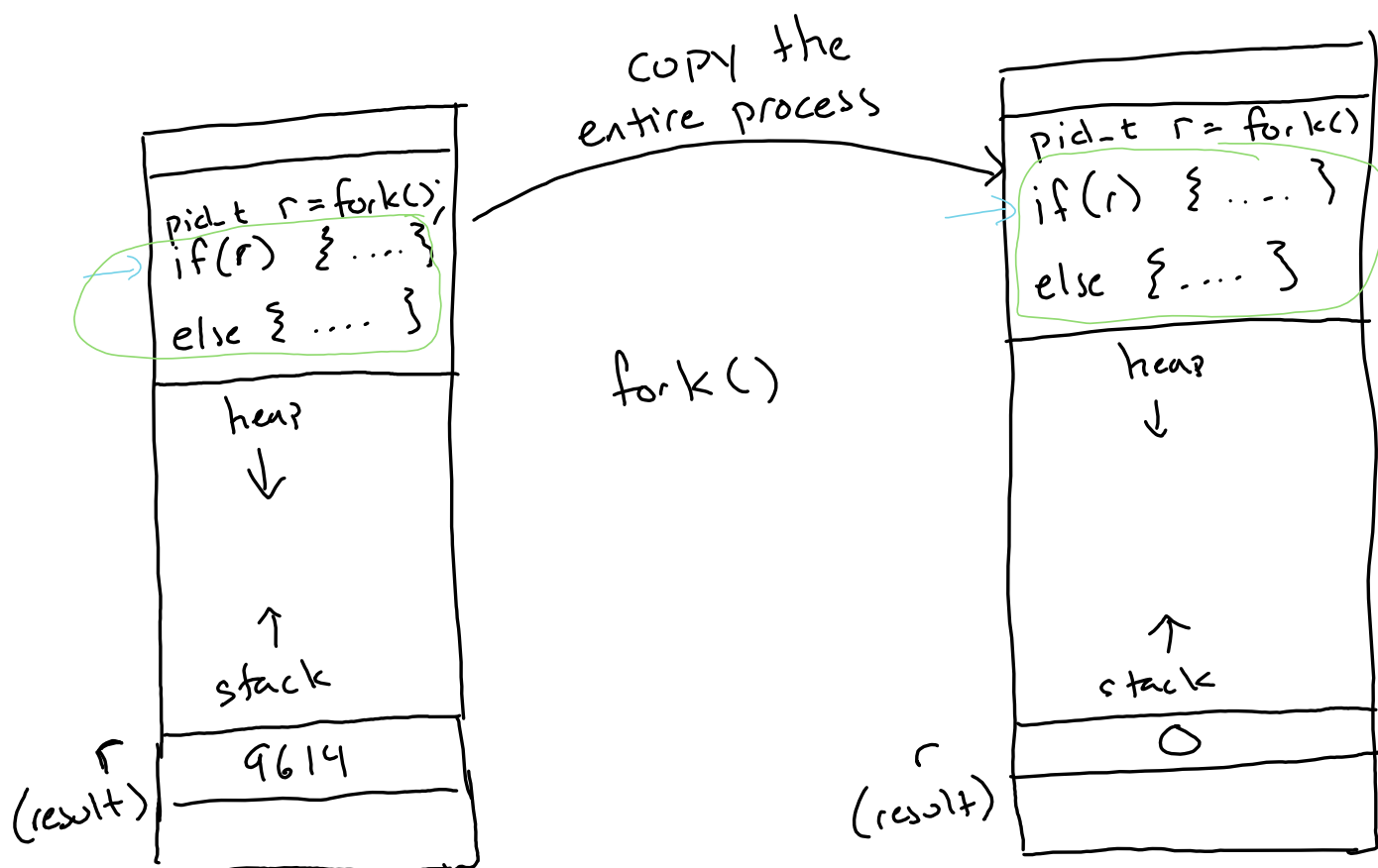
How do we create and manage processes from a program?

pid\_t fork() (creating)

void execvp(cmd, args) (manage/start/control)

What are apps that create + manage processes?

- text editor (proc per file?)
- Youtube? (process for A/V vs. web page) Live Video
- Can we isolate/limit a process (antivirus?)
- bash/terminal (these are C programs!)
- Gradescope autograder / PrairieLearn



fork returns the pid of the copy (eg 9614)

fork returns 0



the code from ls

fresh heap

execvp("ls", args)

fresh stack as if just starting a program