A final evil process and fork

- Topics
  - The perils of recursion
  - Creating new processes: `fork`
- Learning Objectives:
  - Explain the impact of recursion on memory consumption
  - Design ways to limit a process's stack consumption.
  - Explain what the fork system call does from an application programming perspective.
  - Explain what the fork system call does from the operating system perspective.

Recursion: Friend or foe?

```
unsigned f_helper(unsigned i);
unsigned f(unsigned i) {
    if (i == 0)
        return i;
    else
        return f_helper(i) + i;
}
unsigned f_helper(unsigned i) {
    return f(i - 1);
}
void process_main(void) {
    app_printf(0, "Hello from process %d\n", sys_getpid());
    for (unsigned i = 0; i < 1000; ++i)
        app_printf(0, "f(%u) == %u\n", i, f(i));
spinloop: goto spinloop;
}
```

Screen capture

- The program we just looked at is in p-recurse.c.
- What happens when we run it?
- How can we fix it?

Where do Processes Come From?

- How does Weensy create processes?

Where do Processes Come From?

- How does Weensy create processes?
  - Hand craft the process
  - Create a process structure (struct proc).
  - Create an address space.
  - Load the program into the address space.

Where do Processes Come From?

- How does Weensy create processes?
  - Hand craft the process
  - Create a process structure (struct proc).
  - Create an address space.
  - Load the program into the address space.
- How do real operating systems create processes?

Process Creation models

- There are two models of process creation:
  1. Single system call to create a new process (Windows model).

     ```
     CreateProcess(name, cmdline, processAttrs,
     threadAttrs, inheritHandles, flags, env, cwd,
     startupInfo, procInfo);
     ```

  2. Copy an existing process (UNIX `fork/exec` model)

     fork();

Tradeoffs

Create process anew

Copy process

Tradeoffs

Create process anew

+ Let’s you run whatever program you want.

- Complicated call – includes all setup parameters.

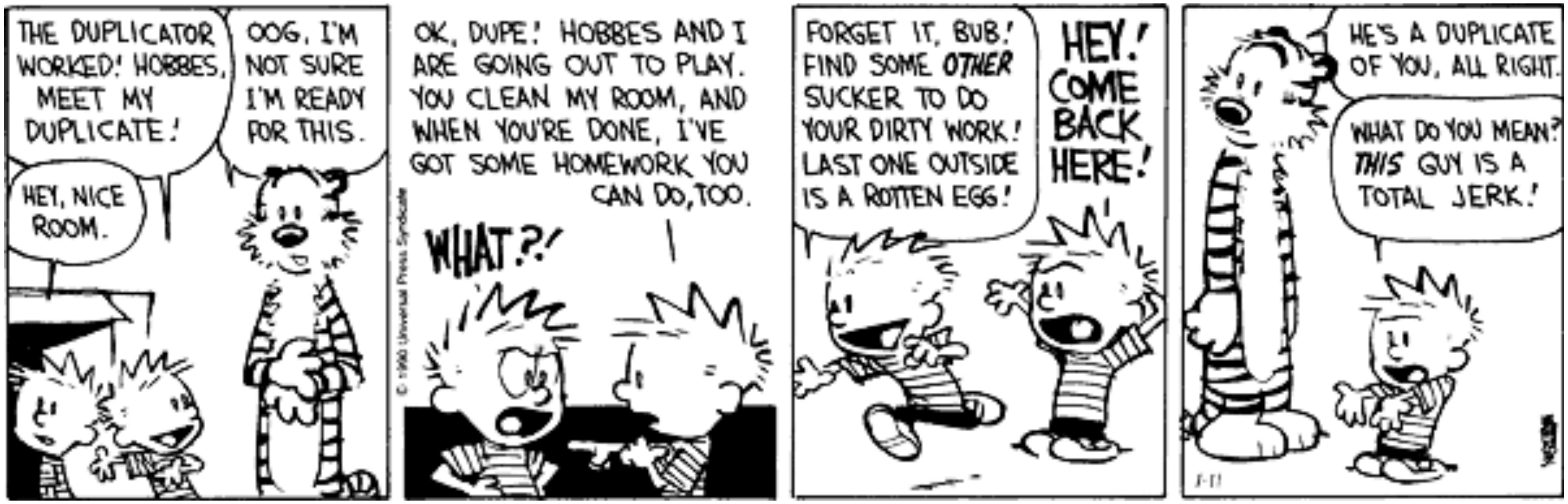
Copy process

- Requires another way to run a different program.

+ Really simple call – setup can be done in the process(es) themselves.

- How do you distinguish the new/old processes?

DUPLICATOR


THE DUPLICATOR WORKED! HOBBES, MEET MY DUPLICATE!
HEY, NICE ROOM.
OOG. I'M NOT SURE I'M READY FOR THIS.
© 1990 Universal Press Syndicate
OK, DUPE! HOBBES AND I ARE GOING OUT TO PLAY. YOU CLEAN MY ROOM, AND WHEN YOU'RE DONE, I'VE GOT SOME HOMEWORK YOU CAN DO, TOO.
WHAT?!
FORGET IT, BUB! FIND SOME OTHER SUCKER TO DO YOUR DIRTY WORK! LAST ONE OUTSIDE IS A ROTTEN EGG!
HEY! COME BACK HERE!
HE'S A DUPLICATE OF YOU, ALL RIGHT.
WHAT DO YOU MEAN? THIS GUY IS A TOTAL JERK!

Creating new processes: `fork`

- System call that copies the calling process, creating a second process that is identical (in all but one regard) to the process that called `fork`.
- We refer to the calling process as the parent and the new process as the child.
- On return from successful `fork`:
  - **Parent: return value is the pid of the child process.**
  - **Child: return value is 0.**
- If the `fork` fails:
  - No child process created.
  - Parent gets return value of -1 (and `errno` is set).

Programming with `fork`

```
#include <unistd.h>
pid_t   ret_pid;

ret_pid = fork();
switch (ret_pid){
        case 0:
                /* I am the child. */
                break;
        case -1:
                /* Something bad happened. */
                break;
        default:
                /*
                 * I am the parent and my child's
                 * pid is ret_pid.
                 */
                break;
 }
```

Full Circle: How do you implement `fork`?

- What does it mean to copy a process?
  - We have to think about the different parts of a process – which ones do we copy?
- Stack?
- Heap?
- Data?
- Text?
- Page tables?
- Registers?
- PID?
- Status?

Screen Capture

- Let’s run fork.
  - What will an strace look like?
- Let’s run fork2.c.
  - How many processes will be created?
- And of course we should run forkbomb.c.
  - What should it do?
  - If you were the OS, what would you do?