Creating Processes

44-550: Operating Systems

Processes

- Normally, we associate a running program with a single process
- Sometimes this behavior isn't what we want
  - Google Chrome
  - HPC applications
  - ...
- Processes can spawn *child processes* that run copies of the code currently running

fork()

- When a processes starts another process, it is said to `fork` the child process
  - so, no, `fork()` is not a utensil
- `fork()` duplicates a process and makes a child process
- The child process has it's own unique PID
  - The PID is unique across the entire system; no other process has the same PID
  - This is my PID. There are many like it, but this one is mine...

```
pid_t fork();
```

- Returns the PID of the child process in the parent, 0 in the child.

fork() Behavior

- The child process contains copies of (almost) all the data stored within the parent process
- Return values differ based on where the code is running.
  - Returns 0 in the child process
  - Returns the PID of the child process in the parent process
  - Returns -1 if the fork fails for some reason
- This is important because we can differentiate between the code based on which process we use.

Common Usage

```
int main()
{
    int cpid = fork();
    if (cpid > 0)
    {
        printf("I'm the parent! Child has PID %d\n", cpid);
    }
    else
    {
        printf("I'm the child!\n");
    }
    return 0;
}
```

`fork` Practice

- Create a global integer variable and initialize it to some value
- In your main function, determine the current PID using the `getpid` function
- Fork, storing the child's PID
- Have the parent print it's pid and it's child's PID
- Have the child process print it's pid (use the `getpid`) function
- Have both processes add a random number (between 0 and 99, inclusive) to the global variable (use the $+=$ operator)
- Have both processes output whether they are child and parent, and the value they get
- Answer the questions on the next slide

Exercise Questions

- Run your program a few times. What do you notice about the PIDs that are output?
- What do you notice about the number that is output to the screen? What does this tell you about the memory for processes? Is it shared? separate? something completely different?
- You may notice that, even if you srand() with time(NULL), both processes are adding the same value to the global integer. How could you make it so the random number generators aren't synced up? (remember that time(NULL)) only has second resolution. What else could you pass to srand that will be unique to each Process. Some kind of IDentifying number, perhaps...