Process? → Thread or Threads

- Address space
- Resource principal
  - CPU (unit of scheduling)
  - RAM (allocation)
  - etc.

- Authority
  - User ID
  - Process hierarchy

- Naming context (e.g., open file descriptors)
- Kernel state
Process context switching

- [Process A executes]
- [Kernel executes]
- [Process B executes]
- [Process A executes]

- Save state to PCB(A)
- Restore from PCB(B)
- Save state to PCB(B)
- Restore from PCB(A)
BOOL CreateProcess(
    in_opt LPCTSTR ApplicationName,
    inout_opt LPCTSTR CommandLine,
    in_opt LPSECURITY_ATTRIBUTES ProcessAttributes,
    in_opt LPSECURITY_ATTRIBUTES ThreadAttributes,
    in BOOL InheritHandles,
    in DWORD CreationFlags,
    in_opt LPVOID Environment,
    in_opt LPCTSTR CurrentDirectory,
    in LPSTARTUPINFO StartupInfo,
    out LPPROCESS_INFORMATION ProcessInformation
);
fork takes no arguments!

```c
pid_t p = fork();
if ( p < 0 ) {
    // Error...
    exit(-1);
} else if ( p == 0 ) {
    // We’re in the child
    execvp("/bin/ls", "ls", NULL);
} else {
    // We’re a parent.
    // p is the pid of the child
    wait(NULL);
    exit(0);
}
```
Context switching with multiple kernel stacks

Monday, October 1, 2018   9:24 AM

Process A

Save to PCB(A)

Decide to switch process

Pick process to run

Switch to Kernel stack B

Restore PCB(B)

Kernel stack A

Kernel stack B

 Kernel stack O
User-space threads
Kernel threads

Monday, October 1, 2018  9:27 AM

[Diagram of kernel threads with labels for user and kernel stacks, CPU 0, and CPU 1]
Microkernels vs. Exokernels

Monday, October 1, 2018 10:23 AM

Diagram showing the differences between microkernels and exokernels in terms of their structure and functionality. The microkernel focuses on scheduling and interprocess communication (IPC), whereas the exokernel includes additional layers for user, threads, file system, virtual machine (VM), network, driver, and protection, multiplexing.