Processes and threads

---

- 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

Monday, October 1, 2018   9:22 AM

[Diagram of process context switching]
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
);
```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
User-space threads

Monday, October 1, 2018  9:24 AM

Diagram of user threads and processes interacting with the kernel and CPUs.
User thread stacks

Monday, October 1, 2018  9:27 AM

Diagram showing the stack layout for different threads, with annotations indicating the initial process stack and a note about memory allocation ("created by malloc()!").
Kernel threads

Monday, October 1, 2018    9:27 AM
Kernel threads stack arrangement

Stack

BSS
Data
Text

Thread 1 stack

Thread 2 stack

Thread 3 stack

BSS
Data
Text

Allocated by the kernel.