Processes

pid = 1000

stack
heap
data/globals
code

file descriptor table:

1	2	3	…

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

pid = 1001

stack
heap
data/globals
code

file descriptor table:

1	2	3	…

saved registers:
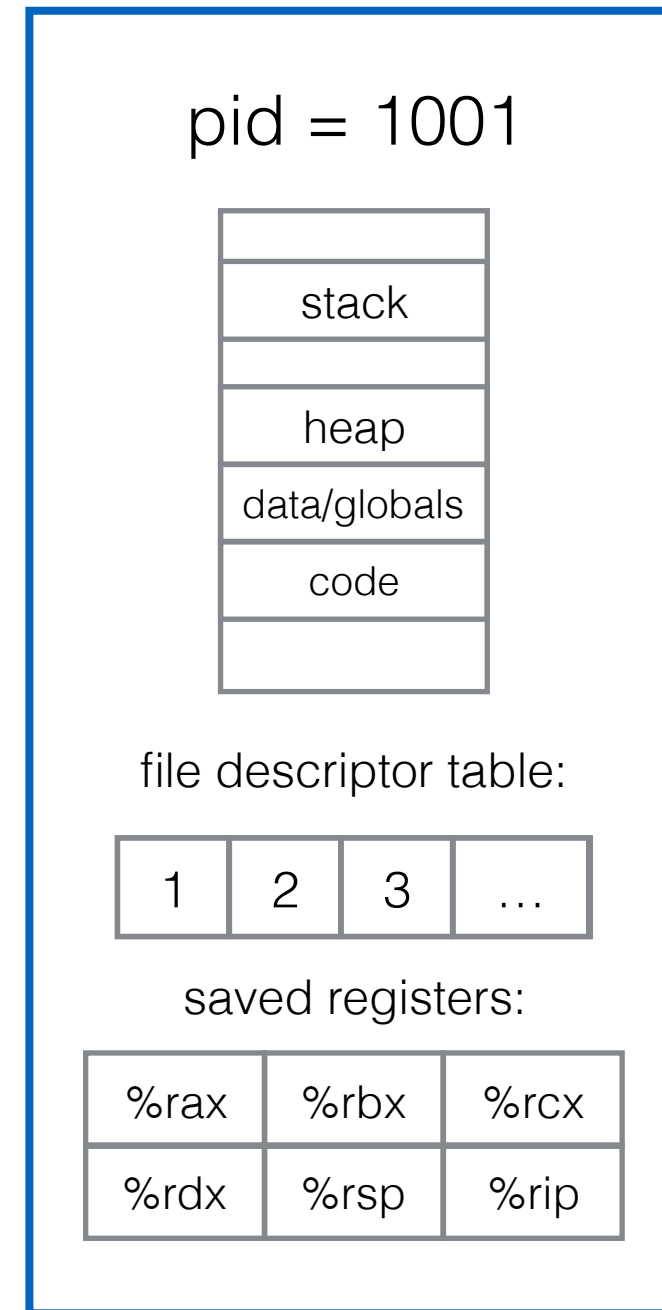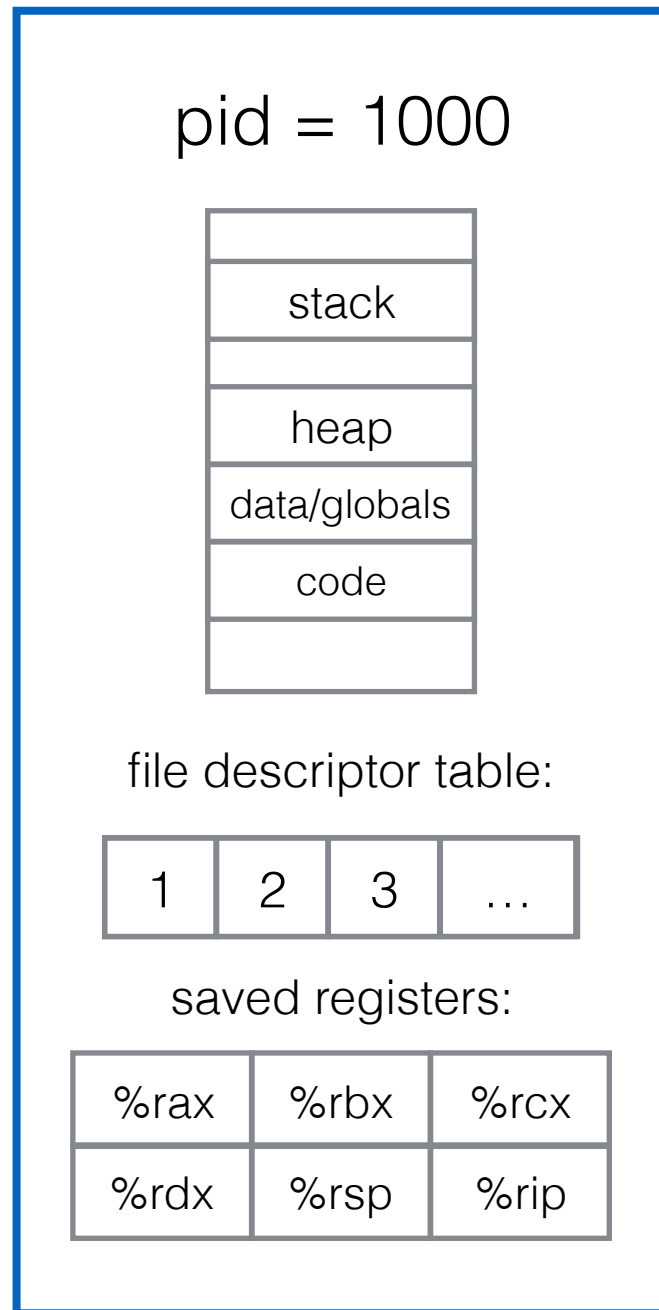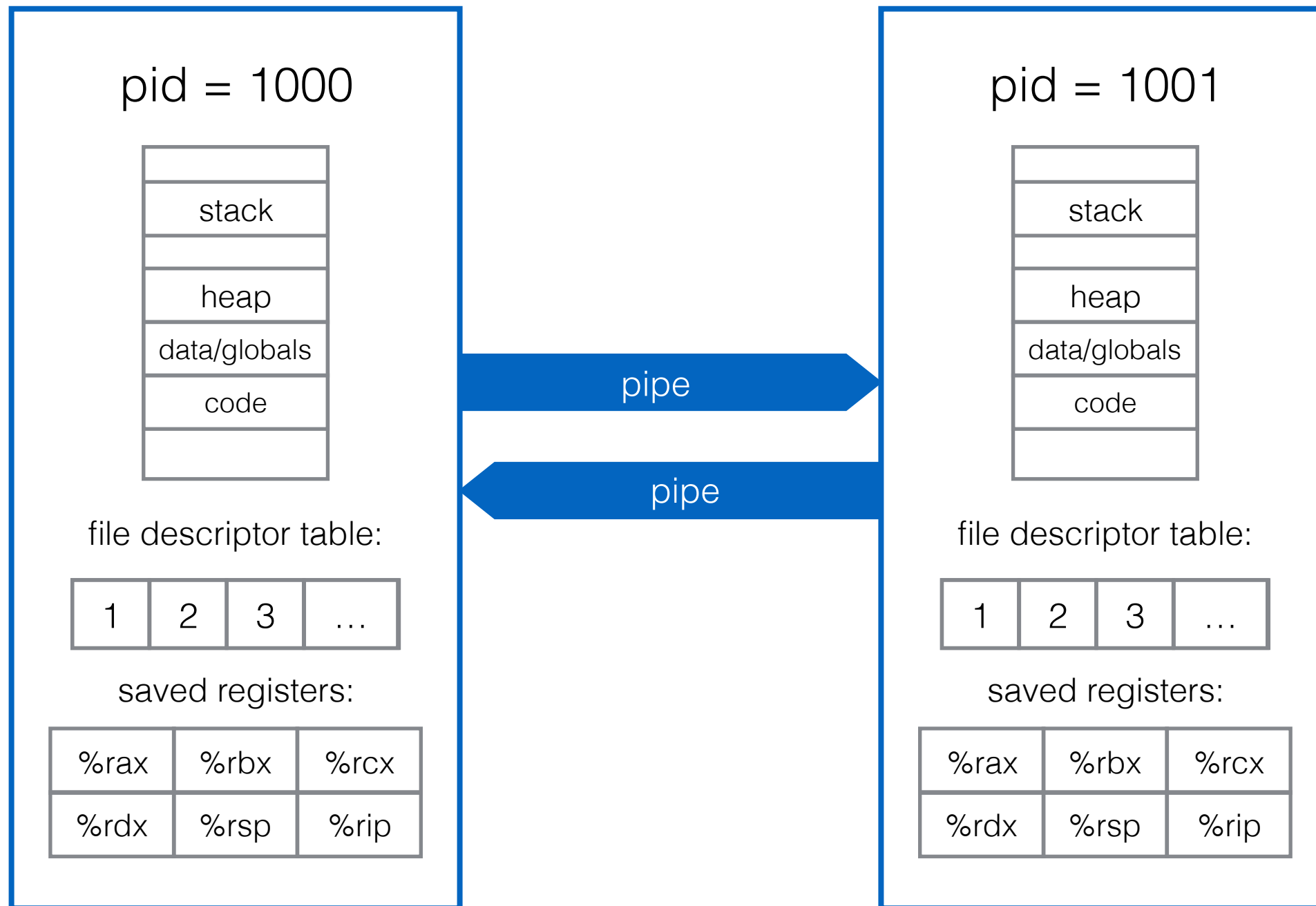
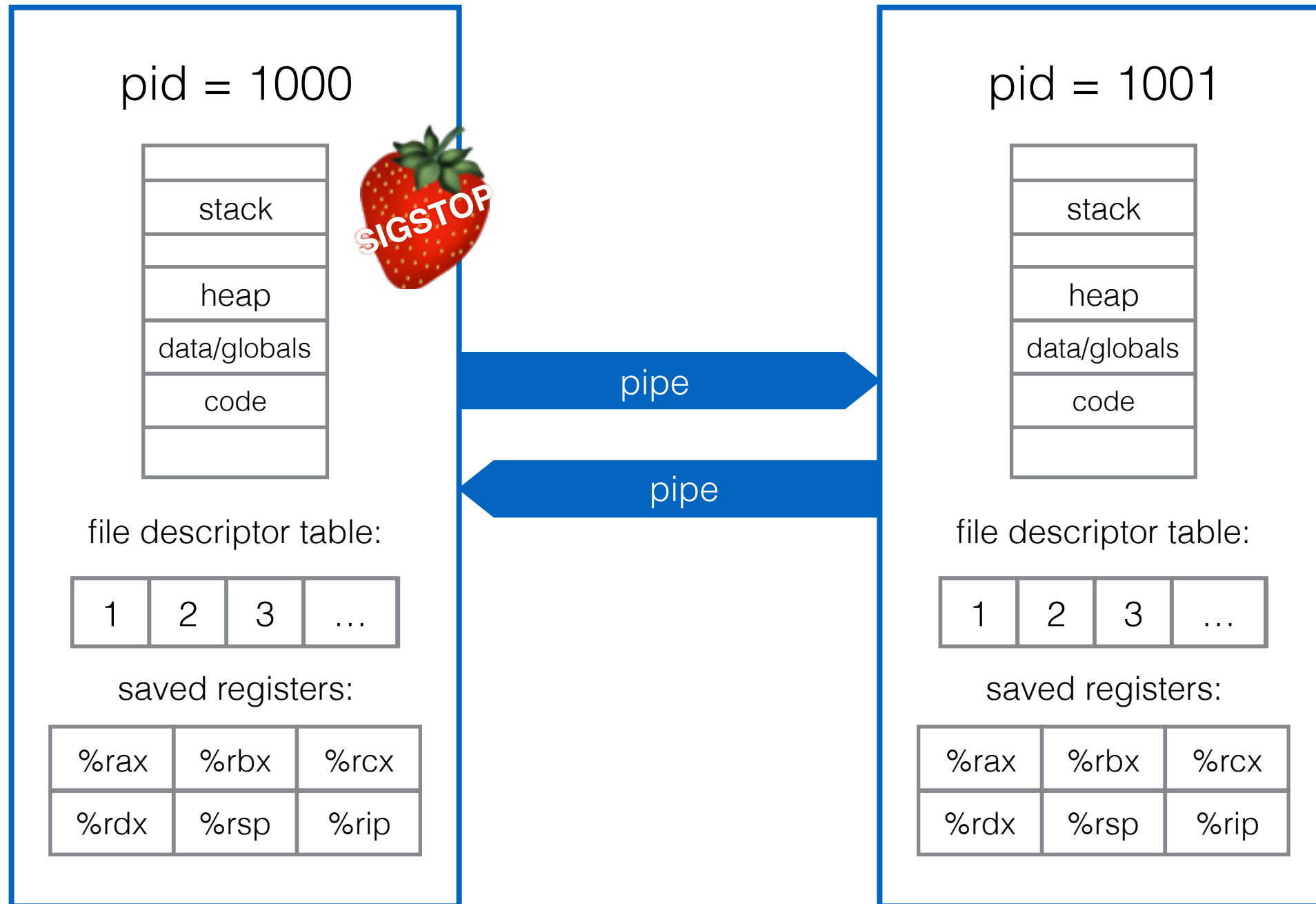| %rax	%rbx	%rcx
%rdx	%rsp	%rip

Every process has its own PID, virtual address space, fd table, registers, signal handlers, etc… Processes are generally *isolated*

Processes

pid = 1000

stack
heap
data/globals
code

pid = 1001

stack
heap
data/globals
code

pipe →

← pipe

file descriptor table:

1	2	3	…

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

file descriptor table:

1	2	3	…

saved registers:

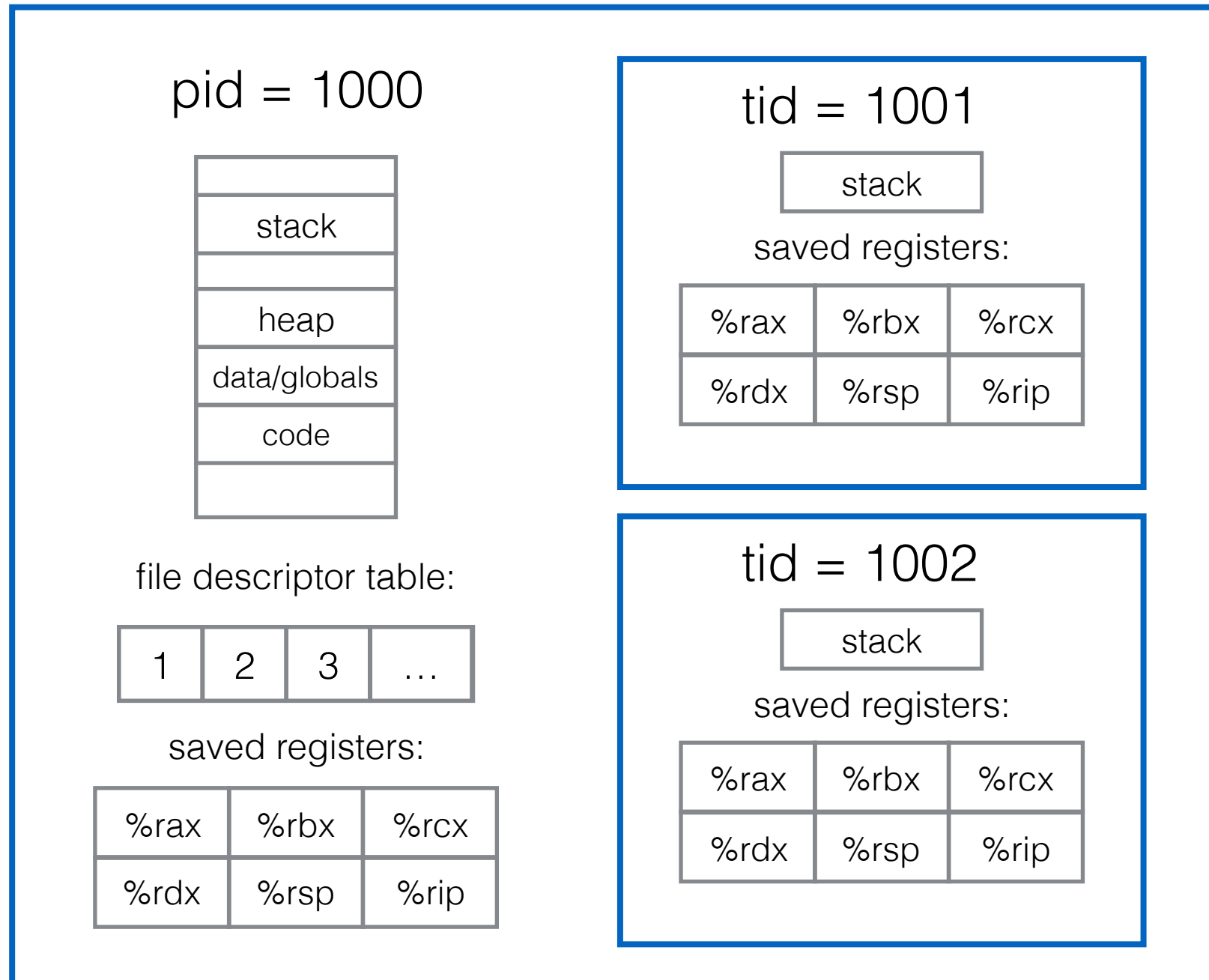| %rax	%rbx	%rcx
%rdx	%rsp	%rip

Processes do *not* share memory (usually), but they can exchange information using pipes

Processes

Processes can synchronize using signals

Threads

pid = 1000		

stack
heap
data/globals
code

file descriptor table:

1	2	3	…

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

tid = 1001

stack

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

tid = 1002

stack

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

Threads are similar to processes; they have a separate stack and saved registers (and a handful of other separated things). But they share most resources across the process

Threads

pid = 1000

stack1
stack2
heap
data/globals
code

file descriptor table:

| 1 | 2 | 3 | ... |

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip

tid = 1001

file descriptor table:

| 1 | 2 | 3 | ... |

saved registers:

%rax	%rbx	%rcx
%rdx	%rsp	%rip
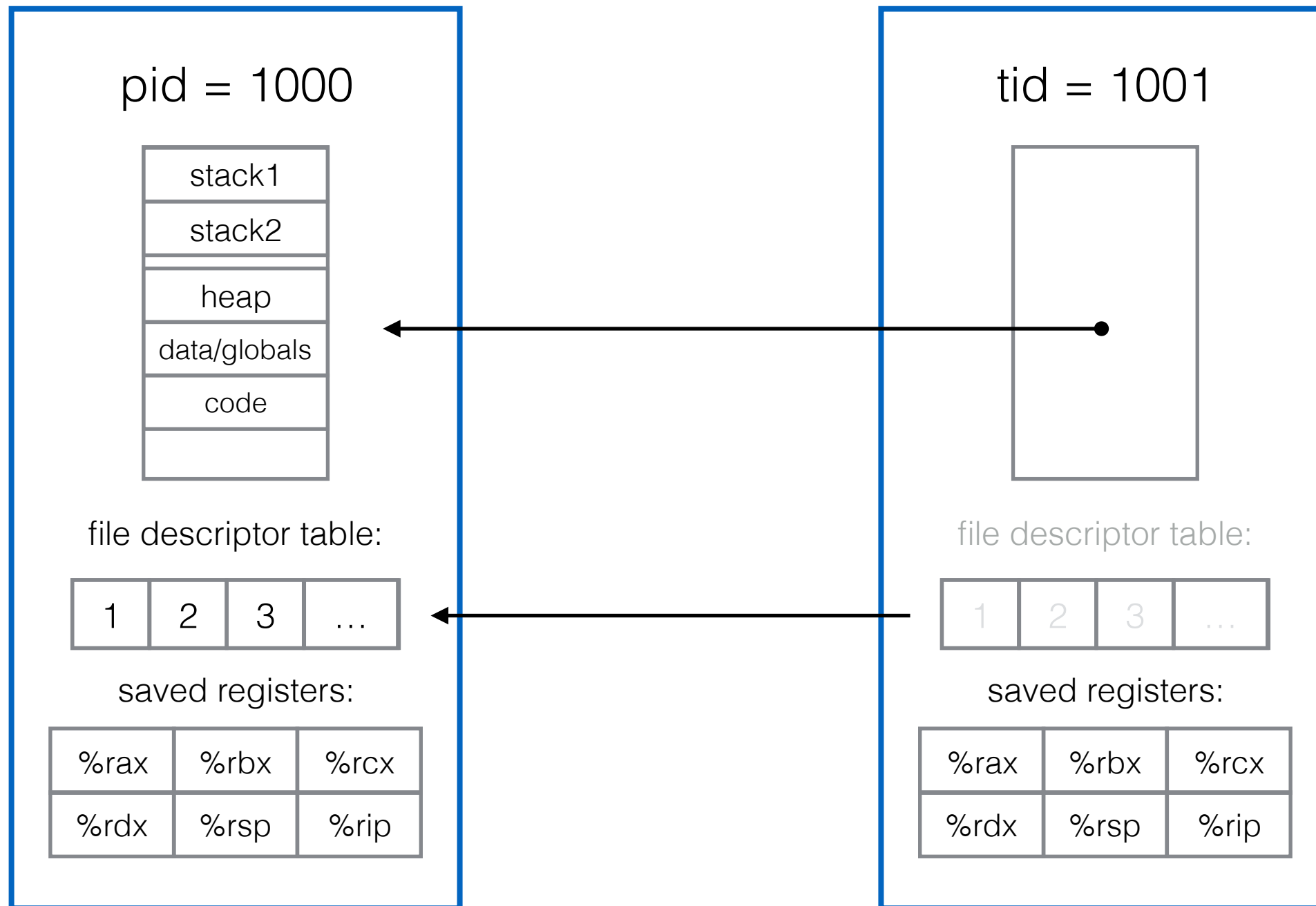
Under the hood, a thread gets its own "process control block" and is scheduled independently, but it is linked to the process that spawned it

What's the difference?

Considerations in designing a browser:

- Speed

- Memory usage

- Battery/CPU usage

- Security, stability

What's the difference?

BROWSER ARCHITECTURE