

# MINIX 3 System Call Flow: User → Kernel → User

## USER PROCESS (Ring 3)

Execute: INT 0x33 / SYSENTER / SYSCALL

## CPU Hardware Transition (Ring 3 → Ring 0)

- Push: EFLAGS, CS, EIP (ESP, SS if privilege ↑)
- Load: Kernel CS:EIP from IDT or MSR
- Switch: Stack to TSS.ESP0 (kernel stack)
- Clear: IF (disable interrupts)

Ring 3→0

## KERNEL ENTRY (mpx.S)

```
ipc_entry_sysenter (220)      [SYSENTER]
ipc_entry_syscall_cpu0-7 (202)   [SYSCALL]
ipc_entry_softint_* (265, 269)  [INT]
```

## SAVE\_PROCESS\_CTX(offset, trap\_style)

Save to proc\_table[current]:  
GPRs: EAX, EBX, ECX, EDX, ESI, EDI, EBP  
Segments: DS, ES, FS, GS  
Special: EIP, ESP, EFLAGS

## C Handler

context\_stop() — Stop user time  
do\_ipc() / kernel\_call() — Handle syscall

## Scheduler

switch\_to\_user() → pick\_proc()  
Select next runnable process  
(May be different process!)

## arch\_finish\_switch\_to\_user(next\_proc)

klib.S:586-651  
IF (next\_proc.CR3 != current\_CR3):  
mov %eax, %cr3 - SWITCH PAGE TABLES

## Context Restore (choose path by trap style)

```
restore_user_context_int (434)    [IRET]
restore_user_context_sysenter (391)  [SYSEXIT]
restore_user_context_syscall (414)   [SYSRET]
```

### Restore from proc\_table[next]

- Reconstruct stack frame for IRET/SYSEXIT/SYSRET
  - Restore all segments (DS, ES, FS, GS, SS, CS)
  - Restore all GPRs from saved state
  - Execute: IRET / SYSEXIT / SYSRET

## CPU Hardware Transition (Ring 0 → Ring 3)

- Restore: EIP, CS, EFLAGS from stack/registers
- Restore: ESP, SS (if privilege ↓)
- Switch: To user stack
- Resume: User-mode execution

Ring 0→3

## USER PROCESS (Ring 3)

Possibly different process than entry!

Resumes with all registers restored

### Legend:

|  |                 |
|--|-----------------|
|  | User Space      |
|  | Kernel Space    |
|  | CPU Hardware    |
|  | Data Structures |

### Abbreviations:

GPR = General Purpose Register

TSS = Task State Segment

IDT = Interrupt Descriptor Table