TI III: Operating & Communication Systems

Subsystems, Interrupts, and System Calls

System Structure,

Flow of Control,

System Library,

POSIX

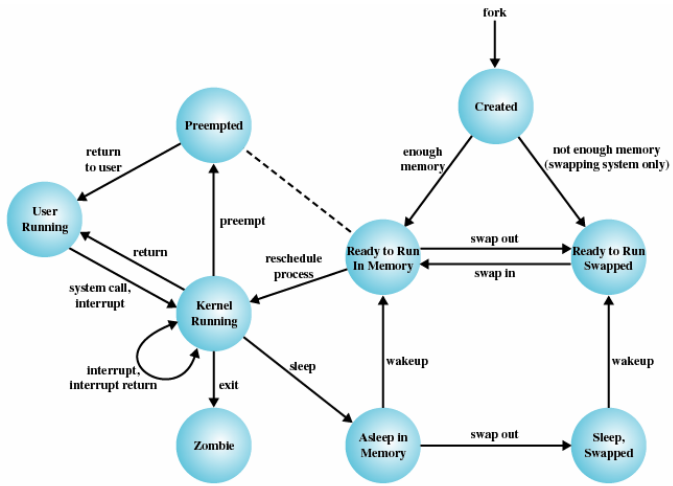


Figure 3.17 UNIX Process State Transition Diagram

Content (1)

1. Introduction and Motivation
   - Tasks
   - Services
   - Virtual Resources
   - Historical Perspective
   - Examples
   - Tools
2. **Subsystems**, Interrupts, and System Calls
   - System Structure
   - Flow of Control
   - System Library
   - POSIX
3. Processes
   1. Definition
   2. Implementation
   3. State Model
4. Memory
5. Scheduling
6. I/O and File System
7. Booting and System Services

Key Elements of an Operating System

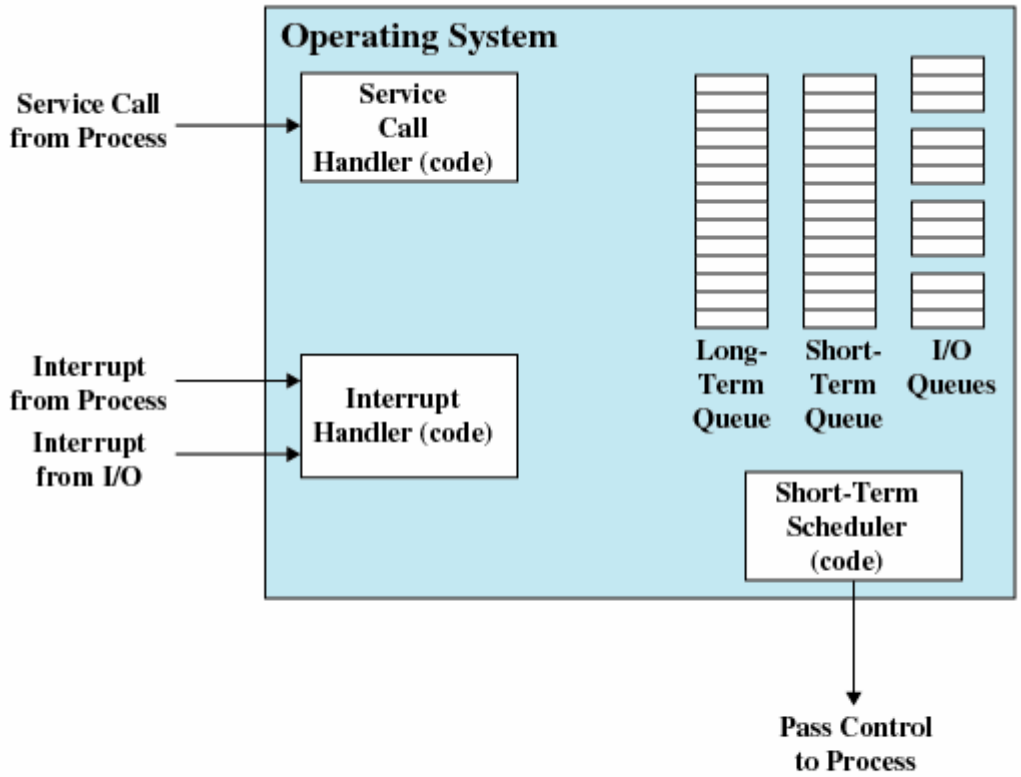


Figure 2.11 Key Elements of an Operating System for Multiprogramming

Hierarchical System View

1. Electronic Circuits
   - Objects: registers, memory cells, logic gates, etc.
   - Operations: clearing a register or reading a memory location
2. Instruction Set
   - Objects: stack, microcode compiler, scalar and vector data
   - Operations: add, subtract, load, store, and branch
3. Procedures
   - Objects: subroutines, call stack
   - Operations: stack pointer, call, return
4. Interrupts
   - Objects: interrupt service routines (ISRs)
   - Operations: call, mask, unmask

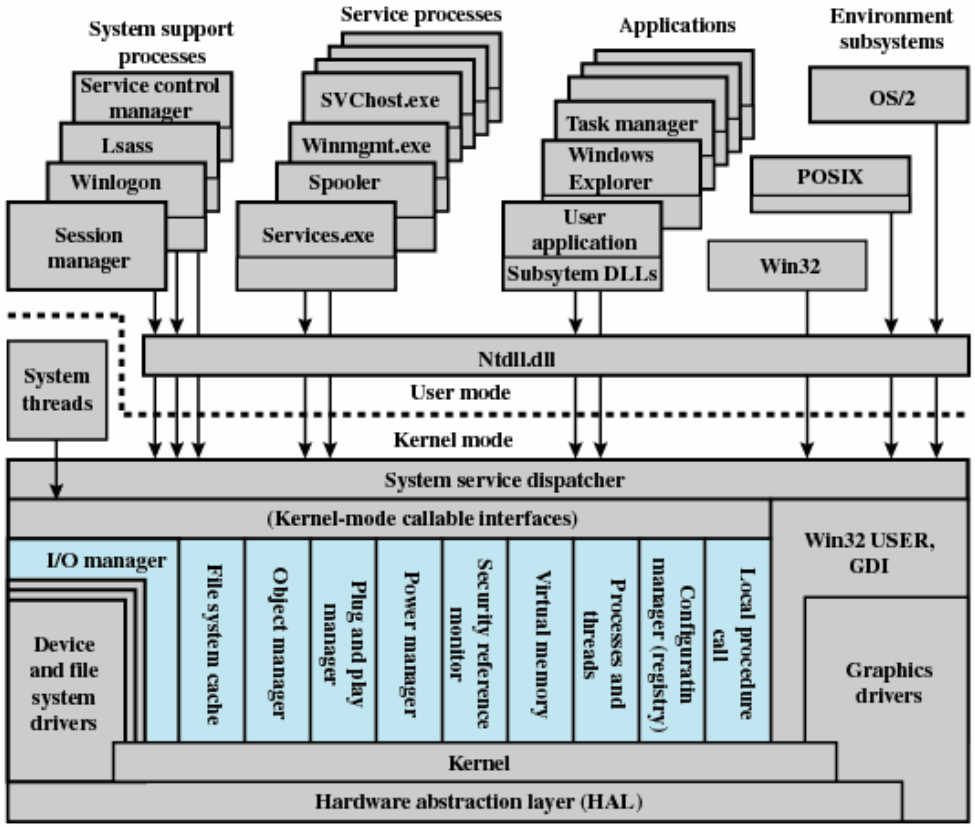
Hierarchical System View

5. Simple Processes
   - Objects: simple processes, semaphore, ready lists
   - Operations: suspend, resume, wait, signal
6. Secondary Storage
   - Objects: blocks of data, device channels
   - Operations: read, write, lock, unlock
7. Virtual Memory
   - Objects: segments, pages
   - Operations: read, write, load
8. Communication
   - Objects: channels
   - Operations: create, delete, open, close, read, write

Hierarchical System View

9. File System
   - Objects: named files
   - Operations: create, delete, open, close, read, write
10. Devices
   - Objects: external devices, e.g. printer, display, and keyboard
   - Operations: open, close, read, write
11. Directories
   - Objects: directories
   - Operations: create, delete, append, remove, search, list
12. User Processes
   - Objects: user processes
   - Operations: create, terminate, suspend, resume
13. Shell
   - Objects: user interface
   - Operations: operations in shell command language

Windows 2000 - Architecture



Lsass = local security authentication server
POSIX = portable operating system interface
GDI = graphics device interface
DLL = dynamic link libraries

Colored area indicates Executive

Figure 2.13 Windows 2000 Architecture [SOLO00]

Windows 2000 – System Components

- Hardware Abstraction Layer (HAL)
  - Mapping of general to platform-specific hardware instructions
  - Isolation of the operation system from different hardware platforms with the goal of portability
  - Encapsulation of system bus, Direct Memory Access (DMA) controller, system timer, memory module, symmetric multiprocessing (SMP)

- Microkernel
  - Scheduling of threads
  - Switching of processes
  - Trap and interrupt handling
  - Synchronization of processes

Windows 2000 - System Components

- Device Drivers
  - Hardware device and file system drivers
  - Mapping of user calls to I/O functions to hardware-specific I/O tasks
- I/O-Manager
  - Single point of access to I/O devices for applications
  - Enforcement of security policies
  - Mapping of logical device names to devices and file systems
- Object manager
  - Management of objects used for representation of resources
  - Common rules for names and security attributes

Windows 2000 - System Components

- Security Control Monitor
  - Access control and auditing of files, processes, address spaces, and I/O devices
- Process manager / thread manager
  - Management of process and thread objects
- LPC Facility (Local Procedure Call)
  - Client/server relationships between applications and subsystems of the operation system
- VMM (Virtual Memory Manager)
  - Mapping of virtual addresses of processes to physical pages in memory

Windows 2000 - System Components

Freie Universität Berlin

- Cache manager
  - Performance improvements in file-based input/output
  - Caching of disk blocks in main memory
  - Scheduling of disk operations

- Windows / graphics module
  - Windows-based graphical user interface
  - Management of display devices

Content (1)

Interrupts

- Mechanisms to facilitate interrupting normal processing of the CPU by another system component (e.g. I/O devices, memory, ...).

- Goal is to increase the efficiency of the CPU by
  - avoiding long waiting intervals due to slow peripheral devices
  - avoiding continuous requests to devices (polling)
  - asynchronous, event-driven flow of control

Flow of control in applications

- Interrupt handling is performed by the operating system in interrupt service routines (ISRs).
- Interrupts temporarily discontinue the currently executing application.

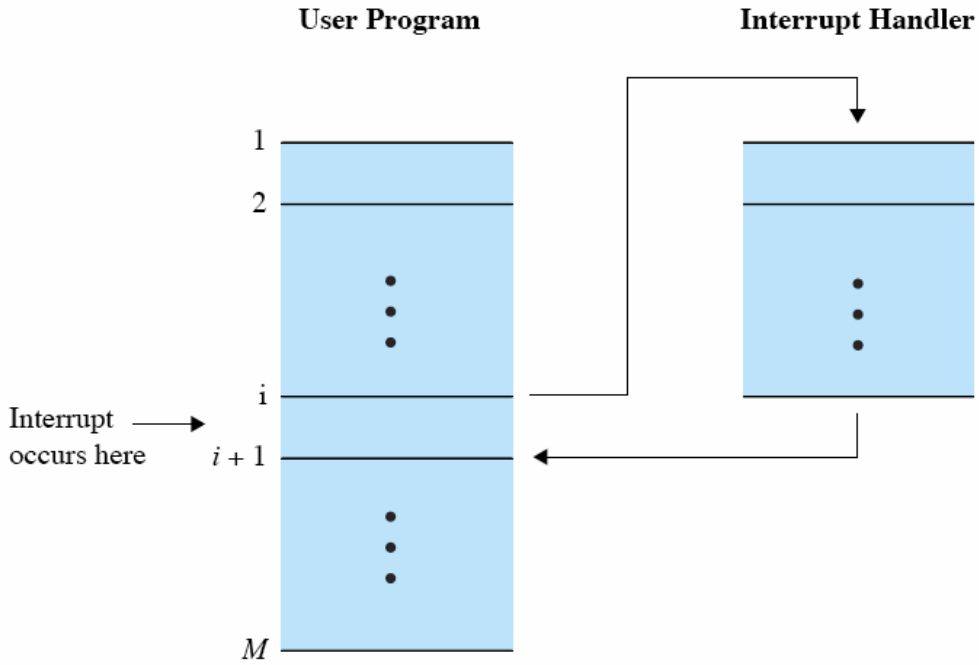
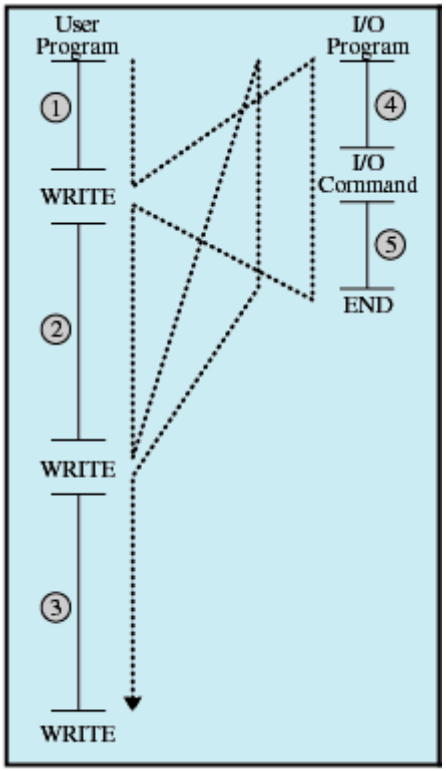
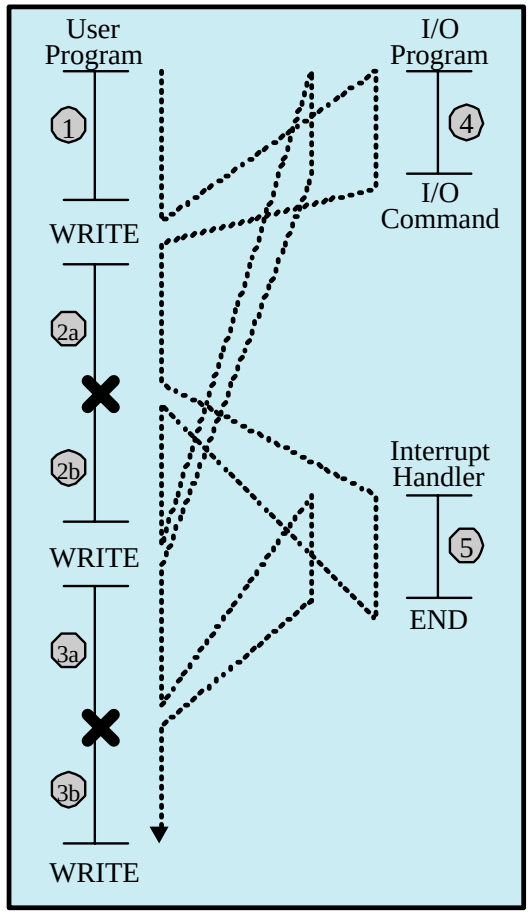


Figure 1.6 Transfer of Control via Interrupts

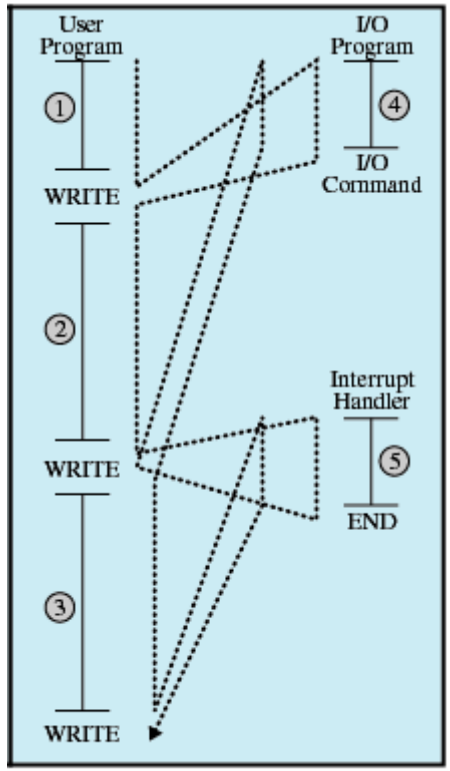
Flow of control with interrupts



(a) No interrupts



(b) Interrupts; short I/O wait



(c) Interrupts; long I/O wait

In-processor interrupt handling

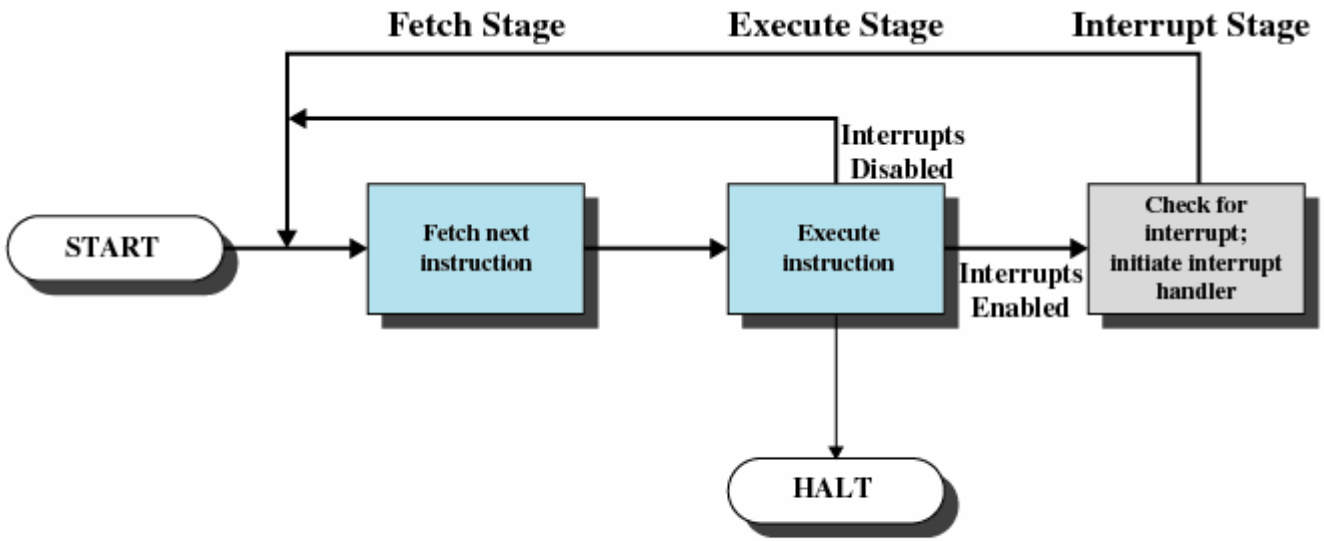


Figure 1.7 Instruction Cycle with Interrupts

Interrupts in hardware and software

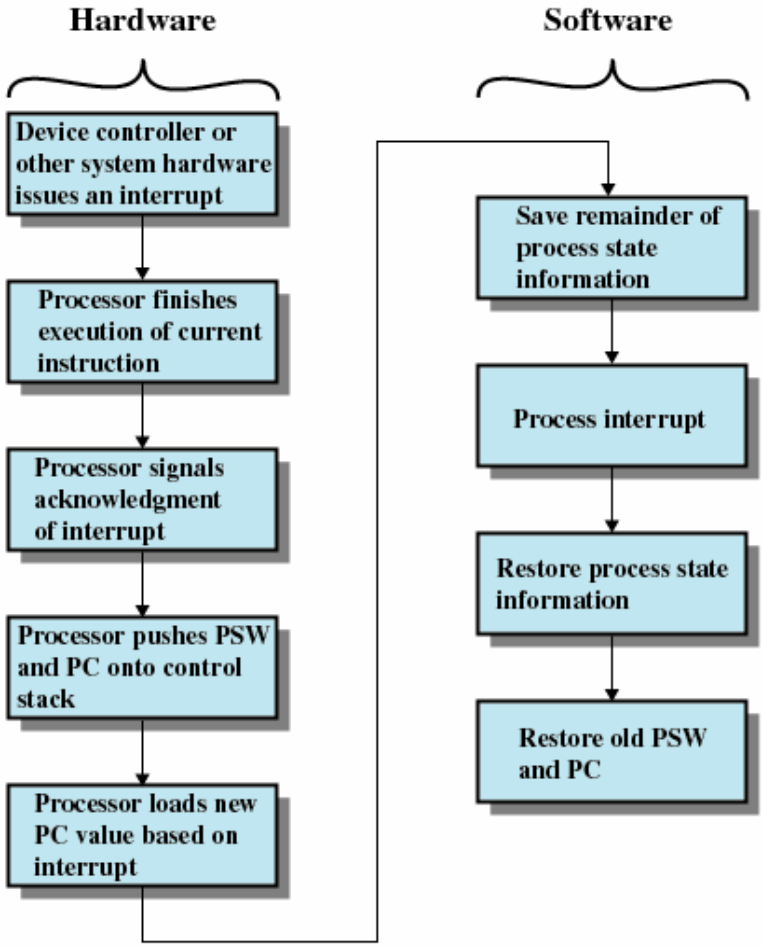


Figure 1.10 Simple Interrupt Processing

Typical steps of interrupt handling

1. I/O device sends interrupt signal to processor.
2. Processor executes current instruction to completion.
3. Processor checks for interrupts. If any
   - processor sends acknowledgement to I/O device,
   - I/O device cancels interrupt signal.
4. Processor prepares to hand control to interrupt service routine.

   Pushes:
   - Processor status word (PSW) of the currently executing process
   - Address of the next instruction to be executed (program counter / PC)

   to the top of the system stack.
5. Load the starting address of the interrupt service routine into the program counter.

Typical steps of interrupt handling

6. Save further process information:
   - Push user-visible processor registers to the system stack

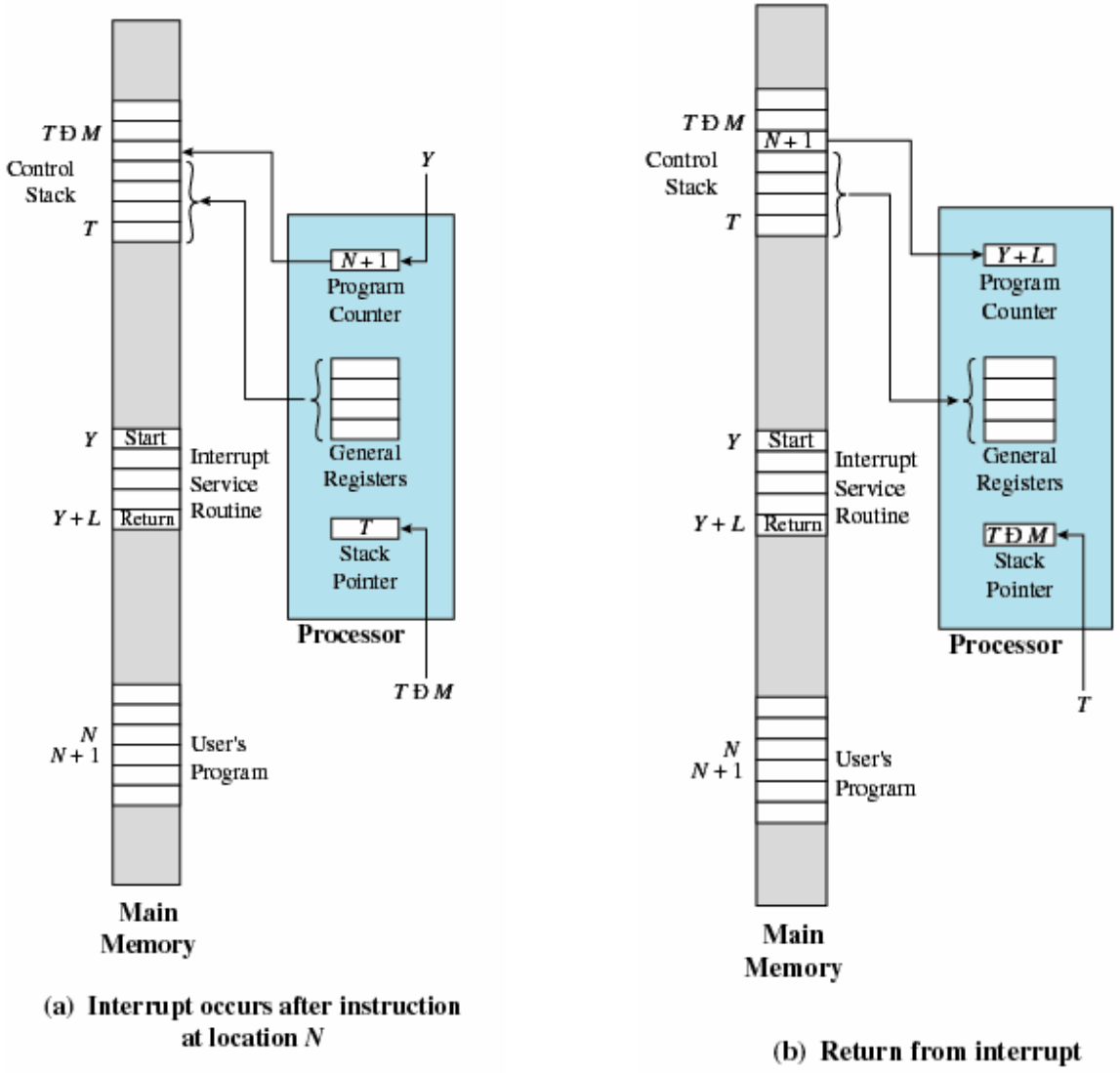
7. Interaction with I/O device

8. Restore process information

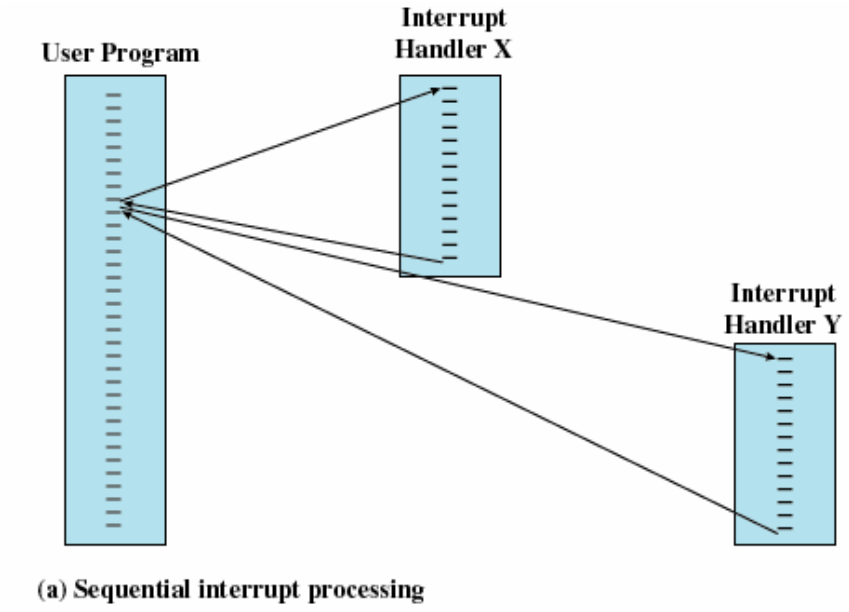
9. Restore processor status word and program counter

➢ The original program transparently continues to execute.

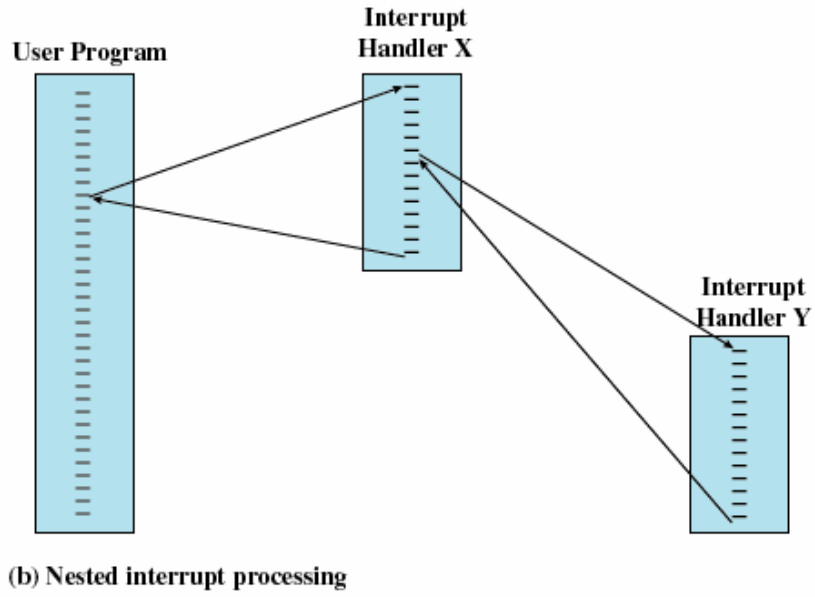
Calling the Interrupt Service Routine



Multiple Interrupts



(a) Sequential interrupt processing



(b) Nested interrupt processing

Types of Interrupt

- Program:
  - May occur during execution of single instructions.
  - Signaled to the current process.
    - Arithmetic overflow, division by zero (SIGFPE)
    - Illegal instruction (SIGILL)
    - Illegal memory access (SIGSEGV)
- Timer:
  - Raised by system timer.
  - Allows operating system to run time-dependant functions.
- I/O:
  - Raised by I/O devices for asynchronous communication, e.g. to handle external events.
- Hardware failure:
  - Raised by hardware malfunction.

Content (1)

1. Introduction and Motivation
   - Tasks
   - Services
   - Virtual Resources
   - Historical Perspective
   - Examples
   - Tools
2. Subsystems, Interrupts, and **System Calls**
   - System Structure
   - Flow of Control
   - System Library
   - POSIX
3. Processes
   1. Definition
   2. Implementation
   3. State Model
4. Memory
5. Scheduling
6. I/O and File System
7. Booting and System Services

System Calls

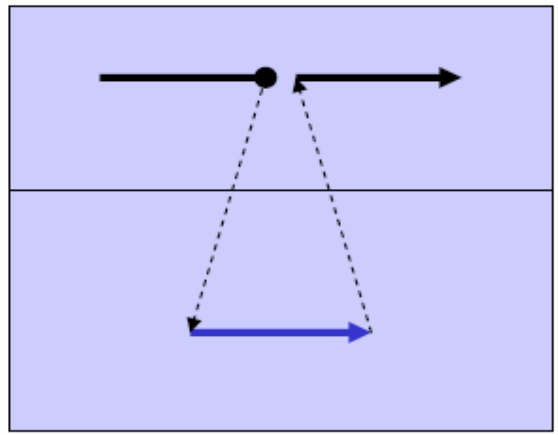
- User applications access system services by calling system calls that are part of the system interface.

- In monolithic kernels:
  1. Subroutine call into operating system
  2. Machine-level instruction “system call”

- In microkernels:
  3. Call of system module / object
  4. Dispatching a task to a system process

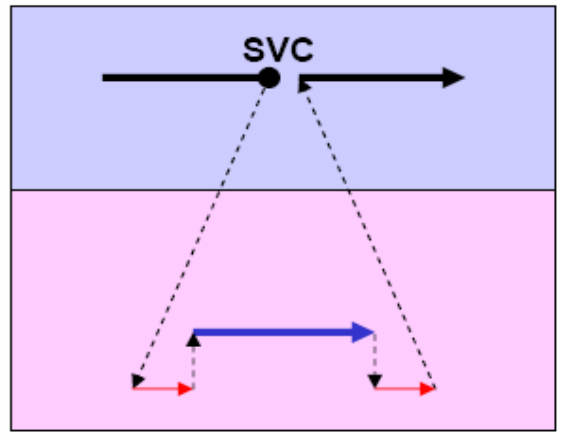
Implementation von Syscalls (1)

- Subroutine call into operating system
  - used in very simple operating systems without separate address spaces
  - Compiler / linker / loader insert call addresses into program
  - Interrupt handling terminates with a simple jump back into program (“return”)
- Example: MS-DOS



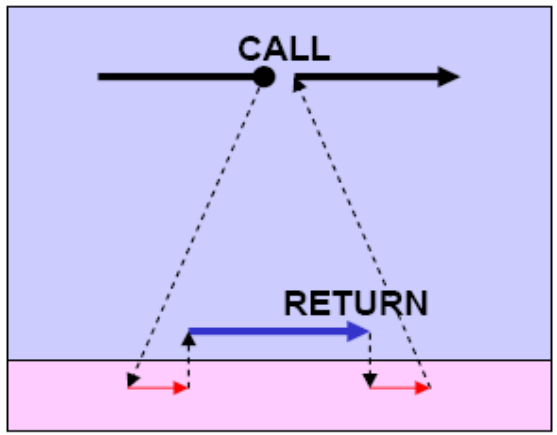
Implementation von Syscalls (2)

- Machine-level instruction “system call” (supervisor call, SVC)
  - Raises trap / exception
  - Interrupt service routine detects cause and branches into corresponding service routine
  - Compiler inserts parameters for system calls
  - Terminates with return
- Example : traditional UNIX



Implementation von Syscalls (3)

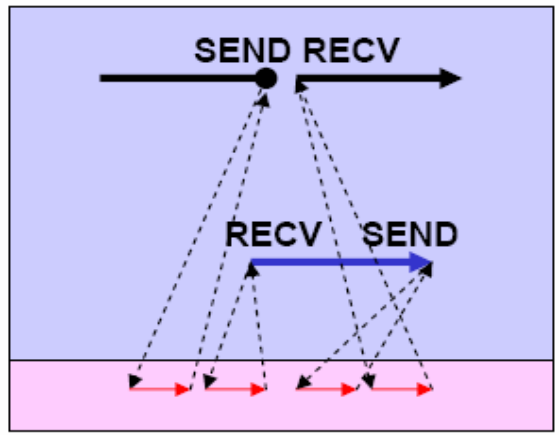
- Call of system module / object
  - Microkernel manages jump into address space of the corresponding system module in response to CALL alarm
  - Return into calling address space with RETURN



Implementation von Syscalls (4)

- Dispatching a task to a system process
  - Microkernel dispatches task to corresponding system process in response to SEND alarm
  - System process receives task with RECV
  - Same method used for delivering result

- Example : Mach, Minix



System Calls and System Library

- System library hides system calls from programmers
- Example **`write()`**:

```
        PUSH   EBX              ; EBX retten
        MOV    EBX,  8(ESP)     ; 1. Parameter
        MOV    ECX, 12(ESP)     ; 2. Parameter
        MOV    EDX, 16(ESP)     ; 3. Parameter
        MOV    EAX, 4           ; 4 steht für „write“
        INT    0x80             ; eigentlicher Systemaufruf
        JBE    DONE             ; kein Fehler
        NEG    EAX              ; Fehlercode
        MOV    errno, EAX
        MOV    EAX, -1
DONE:   POP    EBX              ; EBX wiederherstellen
        RET                     ; Rücksprung
```

- Value -1 in register EAX on error

POSIX

- Portable Operating System Interface for UNIX (POSIX)
- Standard for operating system API (IEEE 1003, ISO/IEC 9945)
- Three main parts:
  - POSIX Kernel APIs
  - POSIX Commands and Utilities
  - POSIX Conformance Testing
- Operating system, that implement POSIX:
  - INTEGRITY, Linux, BSD/OS, A/UX, LynxOS, Mac OS X, MINIX, RTEMS, SkyOS, Windows NT

POSIX Versions

- POSIX.1, Core Services
  - (includes Standard ANSI C)
  - Process Creation and Control
  - Signals
  - Floating Point Exceptions
  - Segmentation Violations
  - Illegal Instructions
  - Bus Errors
  - Timers
  - File and Directory Operations
  - Pipes
  - C Library (Standard C)
  - I/O Port Interface Control
- POSIX.1b, Real-time extensions
  - Priority Scheduling
  - Real-Time Signals
  - Clocks and Timers
  - Semaphores
  - Message Passing
  - Shared Memory
  - Asynch and Synch I/O
  - Memory Locking
- POSIX.1c, Threads extensions
  - Thread Creation, Control, and Cleanup
  - Thread Scheduling
  - Thread Synchronization
  - Signal Handling