UPPSALA UNIVERSITET

Informationsteknologi

Today's class

- Finish operating system overview
- Review of more C

Finish operating system overview

UPPSALA UNIVERSITET

Informationsteknologi

Major Achievements

- Processes
- Memory Management
- Information protection and security
- Scheduling and resource management
- System structure

UPPSALA UNIVERSITET

Informationsteknologi

Processes

- A program in execution
- An instance of a program running on a computer
- The entity that can be assigned to and executed on a processor
- A unit of activity characterized by a single sequential thread of execution, a current state, and an associated set of system resources

Difficulties with Designing System Software

- Improper synchronization
  - Ensure a process waiting for an I/O device receives the signal
- Failed mutual exclusion
- Nondeterminate program operation
  - Program should only depend on input to it, not on the activities of other programs
- Deadlocks

Process

- Consists of three components
  - An executable program
  - Associated data needed by the program
  - Execution context of the program
    - All information the operating system needs to manage the process

Process

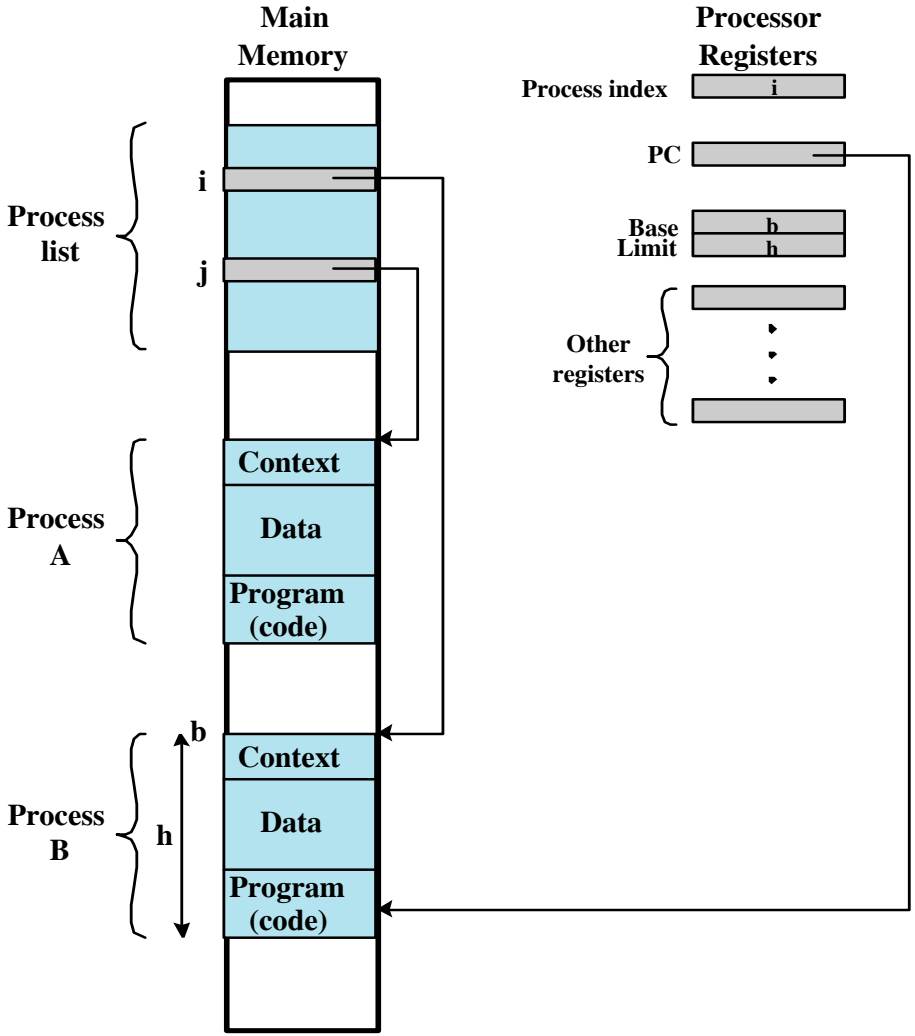


Figure 2.8 Typical Process Implementation

Memory Management

- Process isolation
- Automatic allocation and management
- Support of modular programming
- Protection and access control
- Long-term storage

Virtual Memory

- Allows programmers to address memory from a logical point of view
- No hiatus between the execution of successive processes while one process was written out to secondary store and the successor process was read in

UPPSALA UNIVERSITET

Informationsteknologi

Virtual Memory and File System

- Implements long-term store
- Information stored in named objects called files

UPPSALA UNIVERSITET

Informationsteknologi

Paging

- Allows process to be comprised of a number of fixed-size blocks, called pages
- Virtual address is a page number and an offset within the page
- Each page may be located anywhere in main memory
- Real address or physical address in main memory

Virtual Memory Addressing

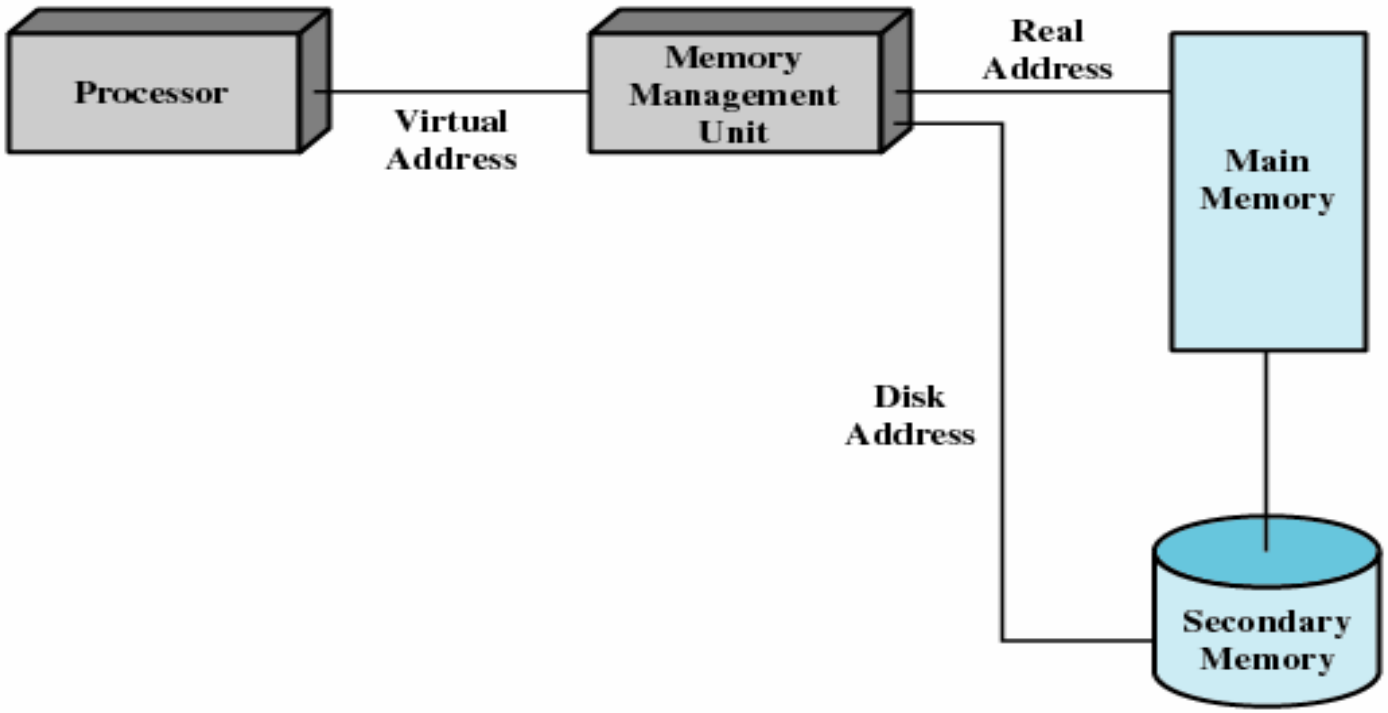


Figure 2.10 Virtual Memory Addressing

UPPSALA UNIVERSITET

Informationsteknologi

Information Protection and Security

- Availability
  - Concerned with protecting the system against interruption
- Confidentiality
  - Assuring that users cannot read data for which access is unauthorized

UPPSALA UNIVERSITET

Informationsteknologi

Information Protection and Security

- Data integrity
  - Protection of data from unauthorized modification
- Authenticity
  - Concerned with the proper verification of the identity of users and the validity of messages or data

UPPSALA UNIVERSITET

Informationsteknologi

Scheduling and Resource Management

- Fairness
  - Give equal and fair access to resources
- Differential responsiveness
  - Discriminate among different classes of jobs
- Efficiency
  - Maximize throughput, minimize response time, and accommodate as many uses as possible

Key Elements of an Operating System

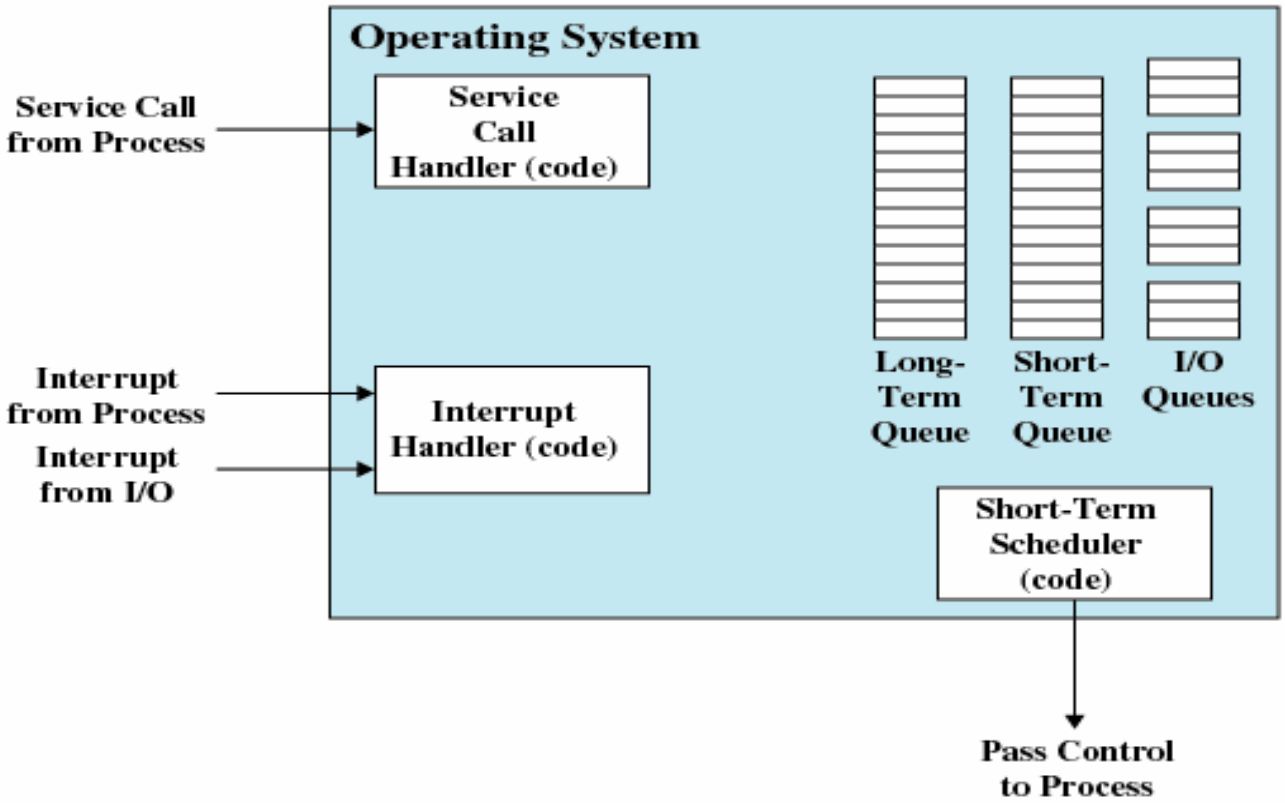


Figure 2.11 Key Elements of an Operating System for Multiprogramming

System Structure

- View the system as a series of levels
- Each level performs a related subset of functions
- Each level relies on the next lower level to perform more primitive functions
- This decomposes a problem into a number of more manageable subproblems

Process Hardware Levels

- Level 1
  - Electronic circuits
  - Objects are registers, memory cells, and logic gates
  - Operations are clearing a register or reading a memory location
- Level 2
  - Processor's instruction set
  - Operations such as add, subtract, load, and store

Process Hardware Levels

- Level 3
  - Adds the concept of a procedure or subroutine, plus call/return operations
- Level 4
  - Interrupts

UPPSALA UNIVERSITET

Informationsteknologi

Concepts with Multiprogramming

- Level 5
  - Process as a program in execution
  - Suspend and resume processes
- Level 6
  - Secondary storage devices
  - Transfer of blocks of data
- Level 7
  - Creates logical address space for processes
  - Organizes virtual address space into blocks

UPPSALA UNIVERSITET

Informationsteknologi

Deal with External Objects

- Level 8
  - Communication of information and messages between processes
- Level 9
  - Supports long-term storage of named files
- Level 10
  - Provides access to external devices using standardized interfaces

Deal with External Objects

- Level 11
  - Responsible for maintaining the association between the external and internal identifiers
- Level 12
  - Provides full-featured facility for the support of processes
- Level 13
  - Provides an interface to the operating system for the user

UPPSALA UNIVERSITET

Informationsteknologi

Modern Operating Systems

- Microkernel architecture
  - Assigns only a few essential functions to the kernel
    - Address spaces
    - Interprocess communication (IPC)
    - Basic scheduling

Modern Operating Systems

- Multithreading
  - Process is divided into threads that can run concurrently
    - Thread
      - Dispatchable unit of work
      - executes sequentially and is interruptable
    - Process is a collection of one or more threads

Modern Operating Systems

- Symmetric multiprocessing (SMP)
  - There are multiple processors
  - These processors share same main memory and I/O facilities
  - All processors can perform the same functions

UPPSALA UNIVERSITET

Informationsteknologi

Modern Operating Systems

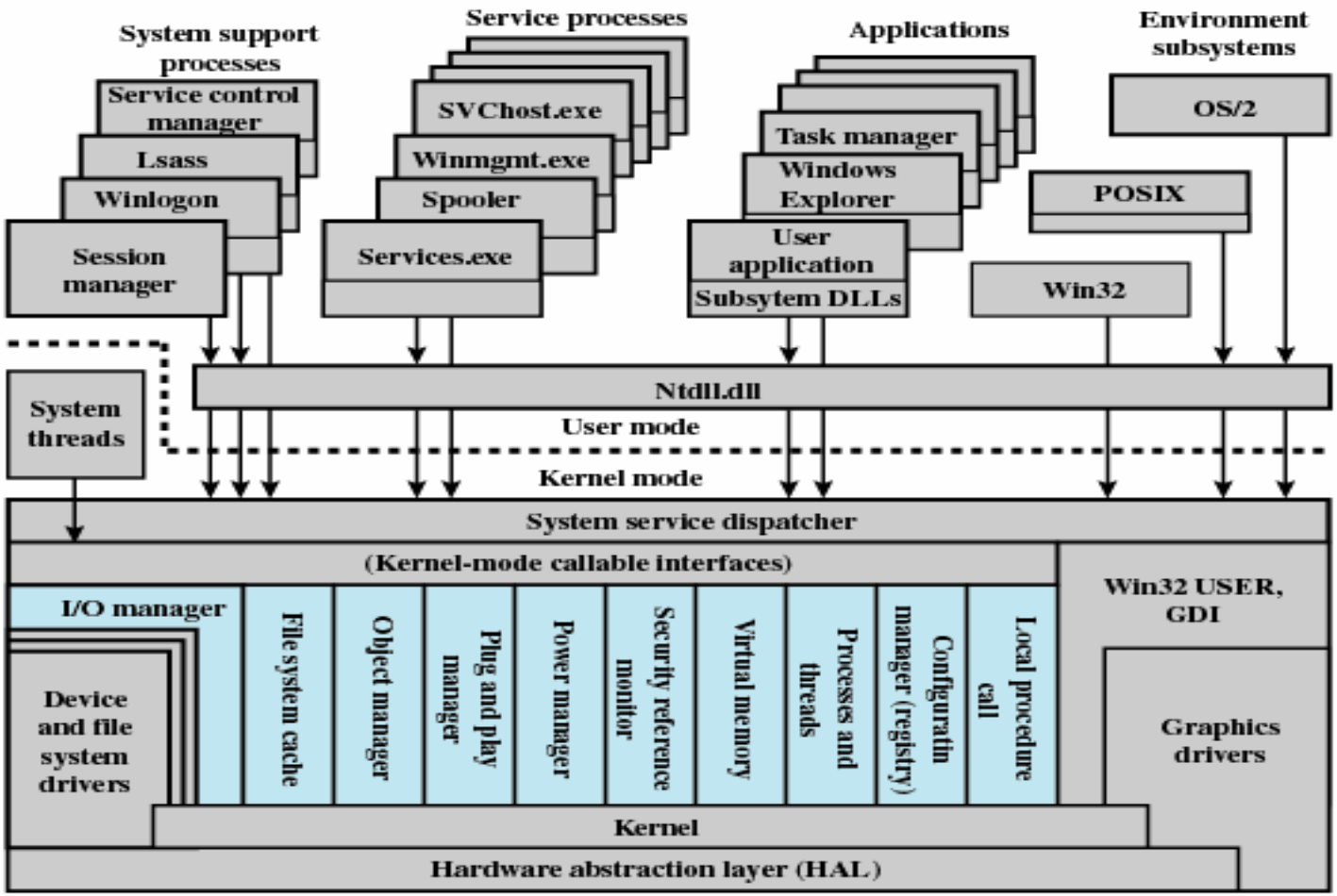
- Distributed operating systems
  - Provides the illusion of a single main memory space and single secondary memory space

Modern Operating Systems

- Object-oriented design
  - Used for adding modular extensions to a small kernel
  - Enables programmers to customize an operating system without disrupting system integrity

Windows Architecture

- Modular structure for flexibility
- Executes on a variety of hardware platforms
- Supports applications written for other operating systems



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]

UPPSALA UNIVERSITET

Informationsteknologi

Operating System Organization

- Modified microkernel architecture
  - Not a pure microkernel
  - Many system functions outside of the microkernel run in kernel mode
- Any module can be removed, upgraded, or replaced without rewriting the entire system

Kernel-Mode Components

- Executive
  - Contains base operating system services
    - Memory management
    - Process and thread management
    - Security
    - I/O
    - Interprocess communication
- Kernel
  - Consists of the most used components

UPPSALA UNIVERSITET

Informationsteknologi

Kernel-Mode Components

- Hardware abstraction layer (HAL)
  - Isolates the operating system from platform-specific hardware differences
- Device drivers
  - Translate user I/O function calls into specific hardware device I/O requests
- Windowing and graphics systems
  - Implements the graphical user interface (GUI)

UPPSALA UNIVERSITET

Informationsteknologi

Windows Executive

- I/O manager
- Cache manager
- Object manager
- Plug and play manager
- Power manager
- Security reference monitor
- Virtual memory manager
- Process/thread manager
- Configuration manager
- Local procedure call (LPC) facility

User-Mode Processes

- Special system support processes
  - Ex: logon process and the session manager
- Service processes
- Environment subsystems
- User applications

UPPSALA UNIVERSITET

Informationsteknologi

Client/Server Model

- Simplifies the Executive
  - Possible to construct a variety of APIs
- Improves reliability
  - Each service runs on a separate process with its own partition of memory
  - Clients cannot not directly access hardware
- Provides a uniform means for applications to communicate via LPC
- Provides base for distributed computing

UPPSALA UNIVERSITET

Informationsteknologi

Threads and SMP

- Operating system routines can run on any available processor
- Different routines can execute simultaneously on different processors
- Multiple threads of execution within a single process may execute on different processors simultaneously
- Server processes may use multiple threads
- Share data and resources between process

Windows Objects

- Encapsulation
  - Object consists of one or more data items and one or more procedures
- Object class or instance
  - Create specified instances of an object
- Inheritance
  - Support to some extent in the Executive
- Polymorphism

UPPSALA UNIVERSITET

Informationsteknologi

UNIX

- Hardware is surrounded by the operating system software
- Operating system is called the system kernel
- Comes with a number of user services and interfaces
  - Shell
  - Components of the C compiler

UNIX

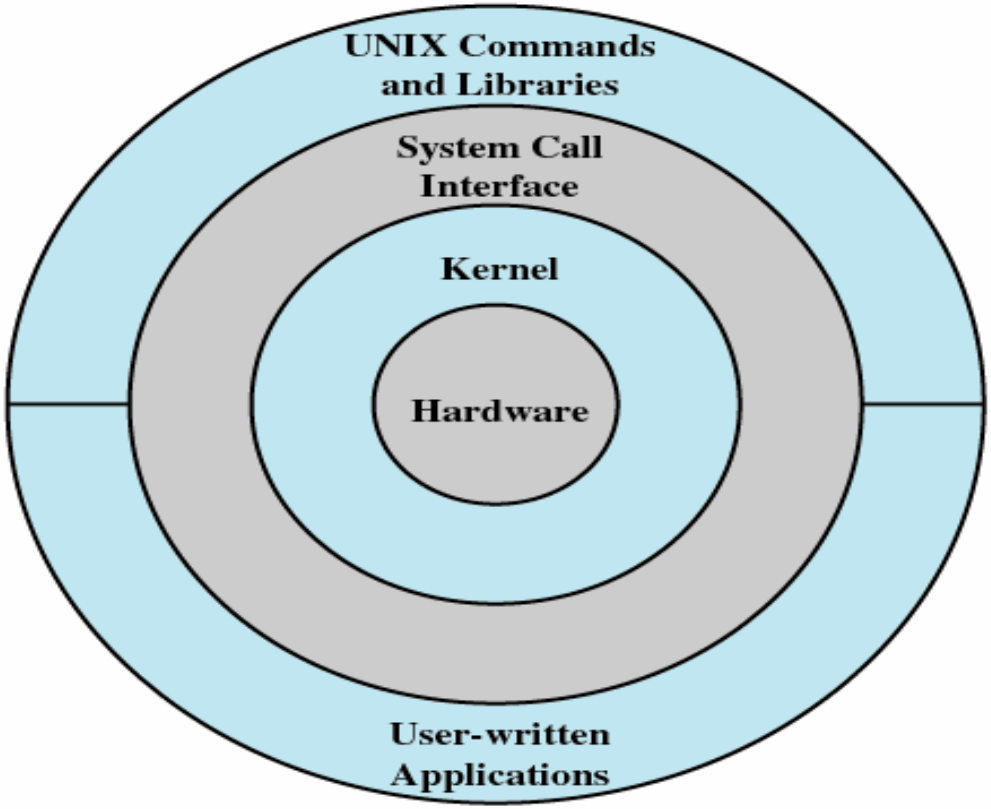


Figure 2.14 General UNIX Architecture

UNIX Kernel

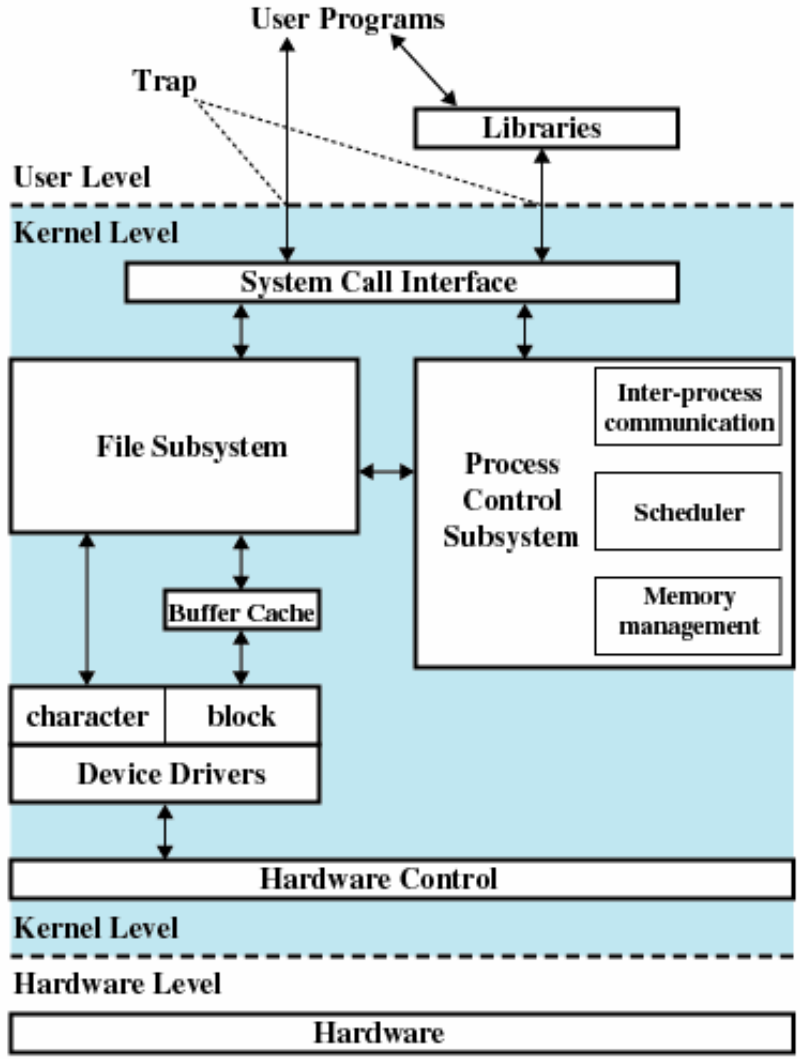


Figure 2.15 Traditional UNIX Kernel [BACH86]

Modern UNIX Kernel

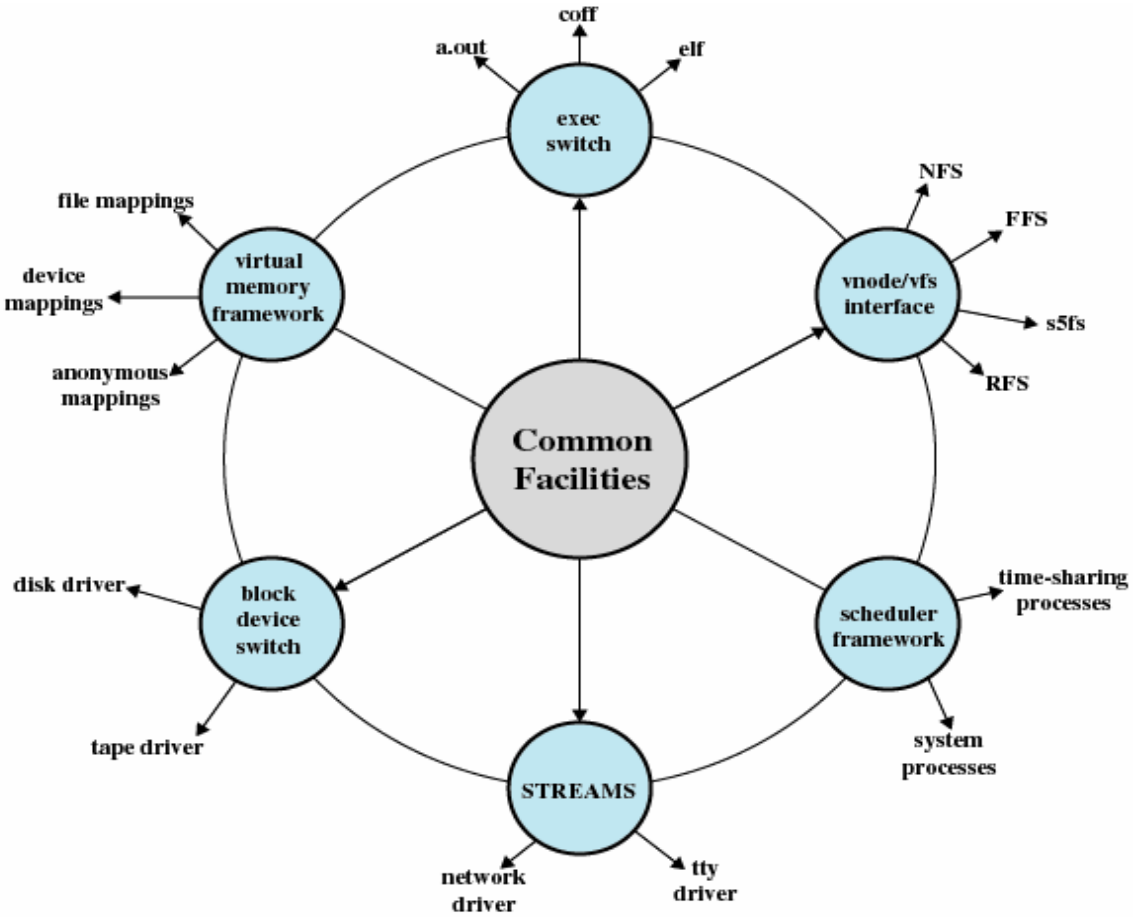


Figure 2.16 Modern UNIX Kernel [VAHA96]

UPPSALA UNIVERSITET

Informationsteknologi

Modern UNIX Systems

- System V Release 4 (SVR4)
- Solaris 9
- 4.4BSD
- Linux

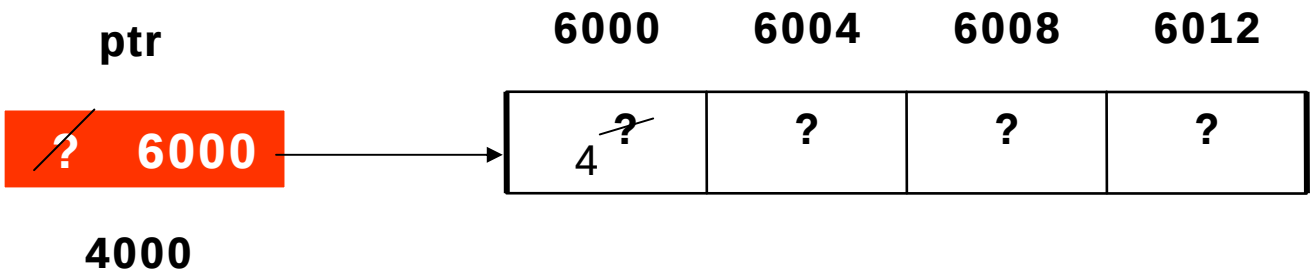
Review of more C

Dynamic memory allocation

- Explicit allocation and de-allocation

```
Example 11
#include <stdio.h>

int
main(int argc, char *argv[])
{
    int *ptr;  /* allocate space to hold an int */

    ptr = (int*)malloc(4 * sizeof(int));

    /* do stuff with the space */
    *ptr=4; //ptr[0] = 4;

    free(ptr);  /* free up the allocated space */
    return 0;
}
```

```
int *ptr;
ptr = (int*)malloc(4 * sizeof(int));
*ptr=4;
```




```
free (ptr);
```

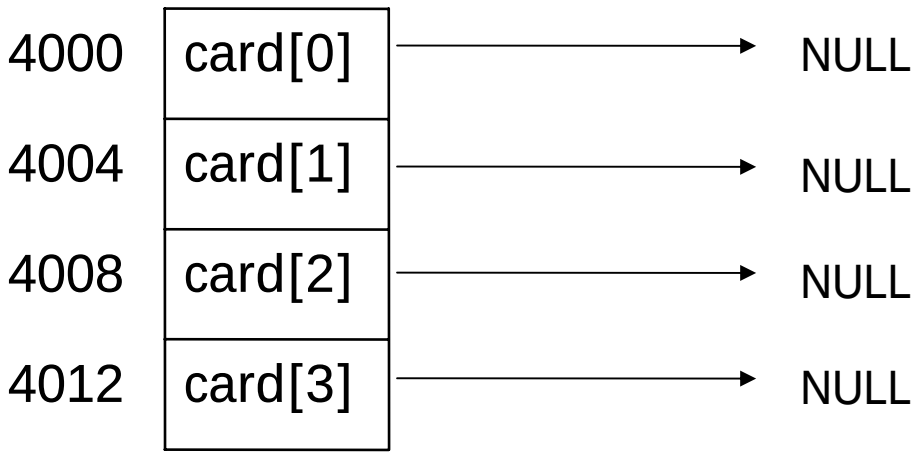
Dynamic array

```
int *ptr, i, size;

printf(“Enter the size of the array”);
scanf(“%d”,&size)

ptr = (int*)malloc( size x sizeof(int) );
for(i=0; i<size; i++){
     ptr[i] = i;
}
```

Array of pointers

- Variable length strings

```
char *card[4]; //card[4] => array of 4 elements
               //char* => element is a pointer to
               //a character.
               //*card[4] => array of 4 pointers
```



```
card[0] = (char*)malloc(6*sizeof(char));
card[1] = (char*)malloc(3*sizeof(char));
```

and so on

Static allocation of a 2D array:

```
char card[4][10];  //waste of space
```

UPPSALA UNIVERSITET

Informationsteknologi

Common errors – memory leak

```
int *ptr, x;
ptr = (int*)malloc(10*sizeof(int));
  //ptr gets space starting at address 3000
ptr = &x;
```

- The space allocated through malloc is no longer available for use by the program.
- Released only when program quits.
- Becomes a problem in large programs where a large number of variables are created and destroyed during the execution of the program.

UPPSALA UNIVERSITET

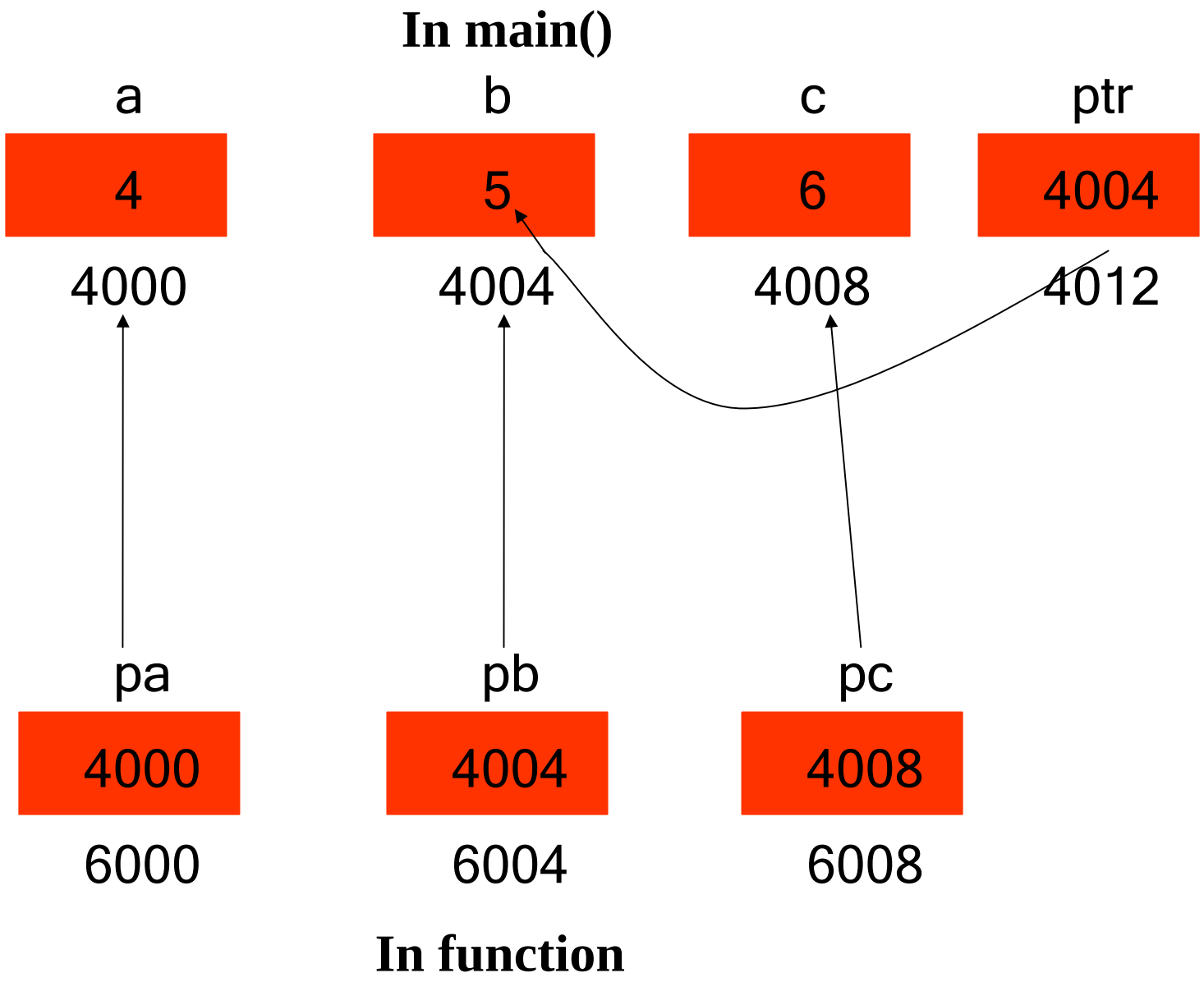
Informationsteknologi

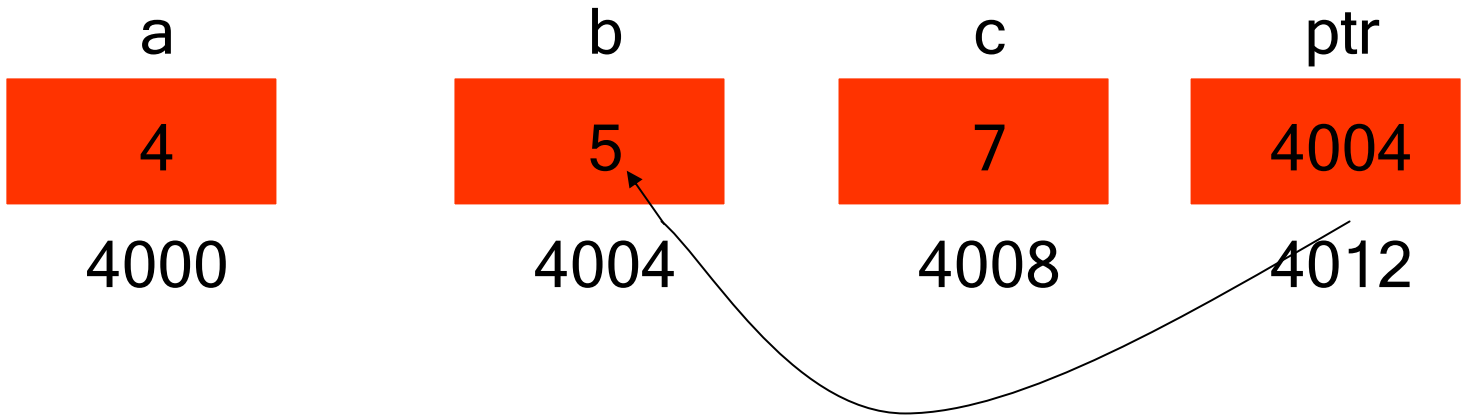
Common errors – dangling pointers

```
int *i, *x;

i = (int*)malloc( 5 x sizeof(int));
x = i;     // both point to the same address.

free(x);  /* both i and x are dangling pointers
             and trying to access either of them
             can cause logical errors
           */

x = NULL;  /* One way to prevent incorrect access */
i = NULL;
```

Functions – pointers as arguments

```
#include <stdio.h>
int sumAndInc(int *pa, int *pb, int* pc);

int
main(int argc, char *argv[])
{
   int a=4, b=5, c=6;
   int *ptr = &b;
   int total = sumAndInc(&a,ptr,&c);
                        /* call to the function */
   printf("The sum of 4 and 5 is %d and c is %p\n", total, c);
}
int sumAndInc(int *pa, int *pb, int *pc ){
                        /* pointers as arguments */
      *pc = *pc+1;      /* return a pointer value */
                        /* NOT *(pc+1) */
      return (*pa+*pb);  /* return by value */
}
```

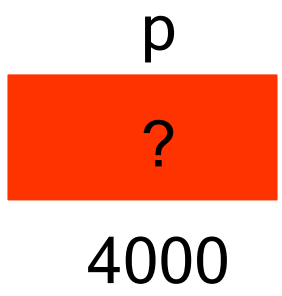

In main()
a
b
c
ptr
4
5
6
4004
4000
4004
4008
4012
pa
pb
pc
4000
4004
4008
6000
6004
6008
In function



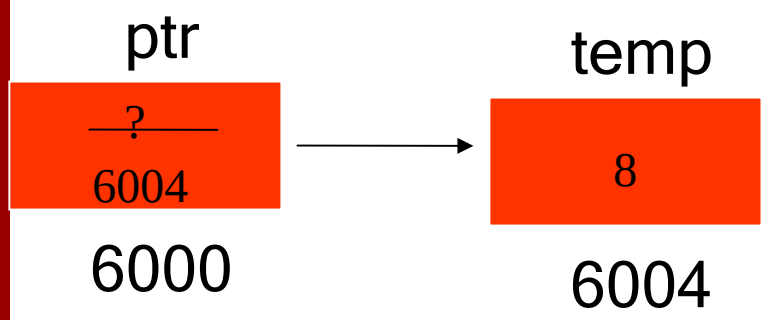
In main() after the function call

What's wrong with this?

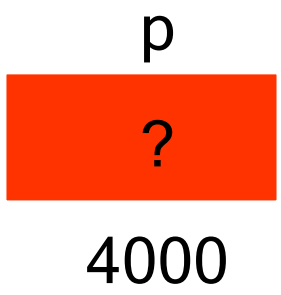
```
#include <stdio.h>

void DoSomething(int *ptr);

int
main(int argc, char *argv[]) {
   int *p;
   DoSomething(p);
   printf("%d", *p);      /* will this work ? */
   return 0;
}

void DoSomething(int *ptr){ /* passed and returned by
                                        reference */
  int temp= 5+3;
  ptr = &(temp);
}

/* compiles correctly, but gives incorrect output */
```



In main()



In the function



In main() after the function call