UPPSALA UNIVERSITET

Informationsteknologi

Today's class

- Finish review of C
- Process description and control

Finish review of C

Review in class exercise 3

- #1: `game`
  `cPtr is 5004`
- #2: `The value of c is 5000`
  `The value of cPtr is 5000`
  `0  1  2  3`
  `The value of cPtr is 5016`
- #3: (a) `*(ptr+2) = 25;`
  (b) `ptr[2] = 25;`

UPPSALA UNIVERSITET

Informationsteknologi

Functions – Passing and returning arrays

```
#include <stdio.h>

void init_array( int array[], int size ) ;

int
main(int argc, char *argv[] )
{
    int list[5];

    init_array( list, 5);
    for (i = 0; i < 5; i++)
        printf("next:%d", list[i]);
}

void init_array(int array[], int size) { /* why size ? */
          /* arrays ALWAYS passed by reference */
    int i;
    for (i = 0; i < size; i++)
        array[i] = 0;
}
```

Passing/returning a struct

```
/* pass struct by value */
void displayYear_1(struct birthday mybday) {
  printf("I was born in %d\n", mybday.year);
}             /* - inefficient: why ? */

/* pass pointer to struct */
void displayYear_2(struct birthday *pmybday) {
  printf("I was born in %d\n", pmybday->year);
        /* Note: '->', not '.', after a struct pointer*/
}

/* return struct by value */
struct birthday get_bday(void){
  struct birthday newbday;
  newbday.year=1971;  /* '.' after a struct */
  return newbday;
}             /* - also inefficient: why ? */
```

Input/output statements

- fprintf(stdout,"….",…); - buffered output
  - Equivalent to printf("….",…)
- fscanf(stdin,…);
  - Equivalent to scanf(…)
- fprintf(stderr,"…",…); - un-buffered output
  - Use for error messages.
- perror(…);
  - Use to print messages when system calls fail.

Storage classes

- Automatic (default for local variables)
  - Allocate memory only when function is executed
  - e.g. `auto int i;`
- Static
  - Allocate memory as soon as program execution begins
  - Scope is local to the function that declares the variable.
  - Value is retained and space is de-allocated only when program (not function) quits.
  - e.g. `static int i;`

Storage classes

- Register
  - Direct compiler to place variable in a register
  - e.g. `register counter = 1;`
- Extern
  - Default for function names.
  - For a variable shared by two or more files:
    - `int i; //global variable in file 1`
    - `extern int i; //global in files 2, 3, …, n`
  - For a function shared by 2 or more files, place a function prototype at the beginning of the files.

enum – enumerated types

```
#include <stdio.h>
enum month{
   JANUARY,       /* like #define JANUARY 0 */
   FEBRUARY,      /* like #define FEBRUARY 1 */
   MARCH          /* … */
};

In main:
enum month birthMonth;
if(birthMonth = = JANUARY){…}

/* alternatively, …. */
enum month{
   JANUARY=1,     /* like #define JANUARY 1 */
   MARCH=3,      /* like #define MARCH 3 */
   FEBRUARY=2,          /* … */
};
```

Process Description and Control

UPPSALA UNIVERSITET

Informationsteknologi

Requirements of an Operating System

- Interleave the execution of multiple processes to maximize processor utilization while providing reasonable response time
- Allocate resources to processes
- Support interprocess communication and user creation of processes

UPPSALA UNIVERSITET

Informationsteknologi

Concepts

- Computer platform consists of a collection of hardware resources
- Computer applications are developed to perform some task
- Inefficient for applications to be written directly for a given hardware platform
- Operating system provides a convenient to use, feature rich, secure, and consistent interface for applications to use
- OS provides a uniform, abstract representation of resources that can be requested and accessed by application

Manage Execution of Applications

- Resources made available to multiple applications
- Processor is switched among multiple applications
- The processor and I/O devices can be used efficiently

UPPSALA UNIVERSITET

Informationsteknologi

Process

- 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 the execution of a sequence of instructions, a current state, and an associated set of system resources

Process Elements

- Identifier
- State
- Priority
- Program counter
- Memory pointers
- Context data
- I/O status information
- Accounting information

Process Control Block

- Contains the process elements
- Created and managed by the operating system
- Allows support for multiple processes

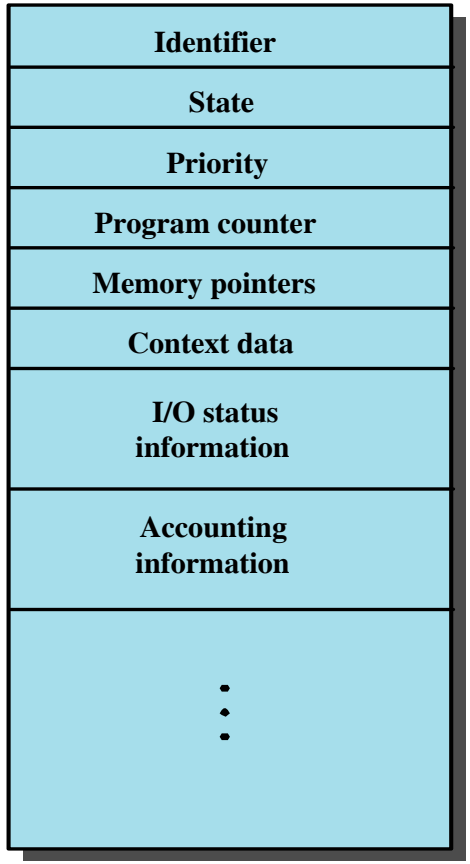


Figure 3.1 Simplified Process Control Block

Example Execution

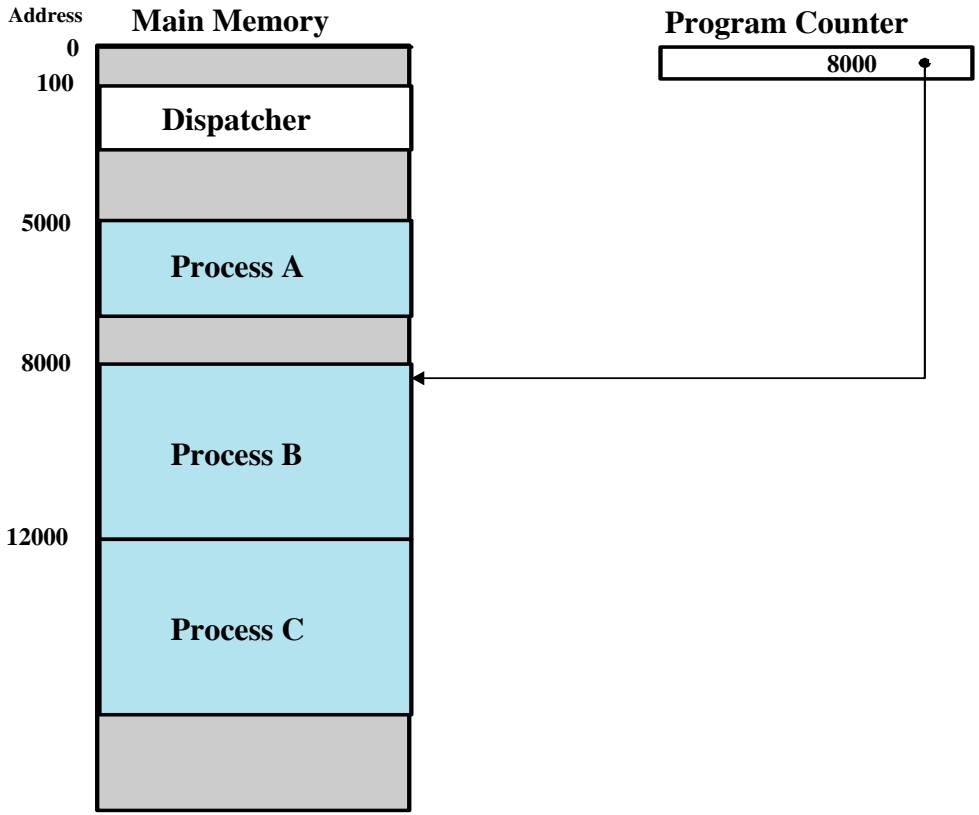


Figure 3.2 Snapshot of Example Execution (Figure 3.4) at Instruction Cycle 13

Trace of Processes

(a) Trace of Process A	(b) Trace of Process B	(c) Trace of Process C
5000	8000	12000
5001	8001	12001
5002	8002	12002
5003	8003	12003
5004		12004
5005		12005
5006		12006
5007		12007
5008		12008
5009		12009
5010		12010
5011		12011

5000 = Starting address of program of Process A
8000 = Starting address of program of Process B
12000 = Starting address of program of Process C

Figure 3.3 Traces of Processes of Figure 3.2

1	5000	27	12004
2	5001	28	12005
3	5002	----------	Time out
4	5003	29	100
5	5004	30	101
6	5005	31	102
----------	Time out	32	103
7	100	33	104
8	101	34	105
9	102	35	5006
10	103	36	5007
11	104	37	5008
12	105	38	5009
13	8000	39	5010
14	8001	40	5011
15	8002	----------	Time out
16	8003	41	100
----------	I/O request	42	101
17	100	43	102
18	101	44	103
19	102	45	104
20	103	46	105
21	104	47	12006
22	105	48	12007
23	12000	49	12008
24	12001	50	12009
25	12002	51	12010
26	12003	52	12011
		----------	Time out

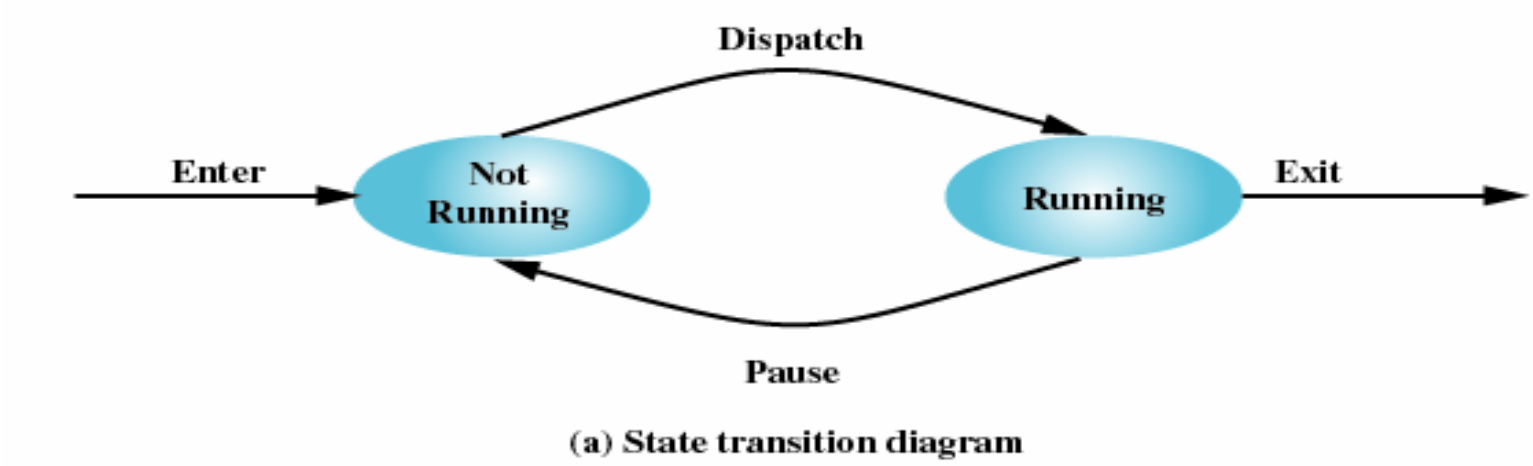
100 = Starting address of dispatcher program

shaded areas indicate execution of dispatcher process;
first and third columns count instruction cycles;
second and fourth columns show address of instruction being executed

Figure 3.4 Combined Trace of Processes of Figure 3.2

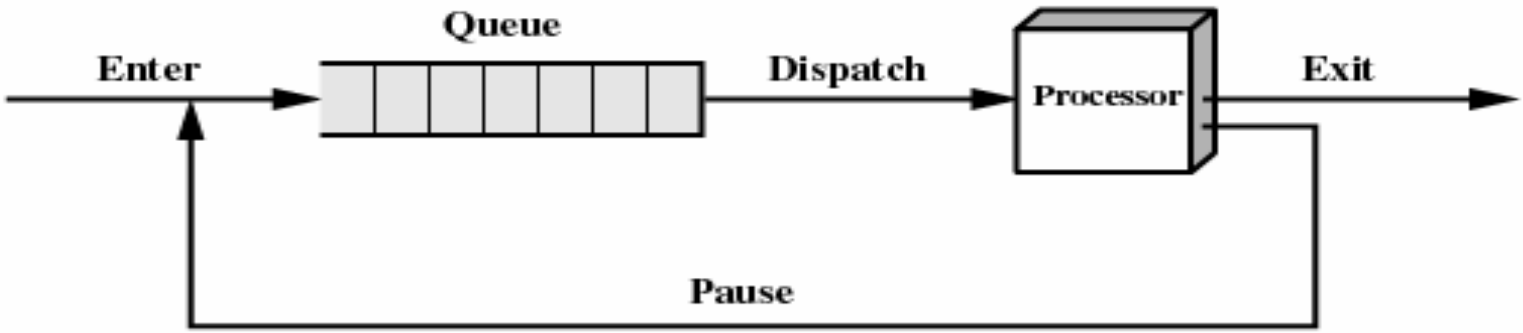
Two-State Process Model

- Process may be in one of two states
  - Running
  - Not-running



(a) State transition diagram

Not-Running Processes in a Queue



(b) Queuing diagram

Process Creation

Table 3.1 Reasons for Process Creation

New batch job	The operating system is provided with a batch job control stream, usually on tape or disk. When the operating system is prepared to take on new work, it will read the next sequence of job control commands.
Interactive logon	A user at a terminal logs on to the system.
Created by OS to provide a service	The operating system can create a process to perform a function on behalf of a user program, without the user having to wait (e.g., a process to control printing).
Spawned by existing process	For purposes of modularity or to exploit parallelism, a user program can dictate the creation of a number of processes.

Process Termination

Table 3.2 Reasons for Process Termination

Normal completion	The process executes an OS service call to indicate that it has completed running.
Time limit exceeded	The process has run longer than the specified total time limit. There are a number of possibilities for the type of time that is measured. These include total elapsed time ("wall clock time"), amount of time spent executing, and, in the case of an interactive process, the amount of time since the user last provided any input.
Memory unavailable	The process requires more memory than the system can provide.
Bounds violation	The process tries to access a memory location that it is not allowed to access.
Protection error	The process attempts to use a resource such as a file that it is not allowed to use, or it tries to use it in an improper fashion, such as writing to a read-only file.
Arithmetic error	The process tries a prohibited computation, such as division by zero, or tries to store numbers larger than the hardware can accommodate.

Process Termination

Table 3.2 Reasons for Process Termination

Time overrun	The process has waited longer than a specified maximum for a certain event to occur.
I/O failure	An error occurs during input or output, such as inability to find a file, failure to read or write after a specified maximum number of tries (when, for example, a defective area is encountered on a tape), or invalid operation (such as reading from the line printer).
Invalid instruction	The process attempts to execute a nonexistent instruction (often a result of branching into a data area and attempting to execute the data).
Privileged instruction	The process attempts to use an instruction reserved for the operating system.
Data misuse	A piece of data is of the wrong type or is not initialized.
Operator or OS intervention	For some reason, the operator or the operating system has terminated the process (for example, if a deadlock exists).
Parent termination	When a parent terminates, the operating system may automatically terminate all of the offspring of that parent.
Parent request	A parent process typically has the authority to terminate any of its offspring.

UPPSALA UNIVERSITET

Informationsteknologi

Processes

- Not-running
  - ready to execute
- Blocked
  - waiting for I/O
- Dispatcher cannot just select the process that has been in the queue the longest because it may be blocked

A Five-State Model

- Running
- Ready
- Blocked
- New
- Exit

Five-State Process Model

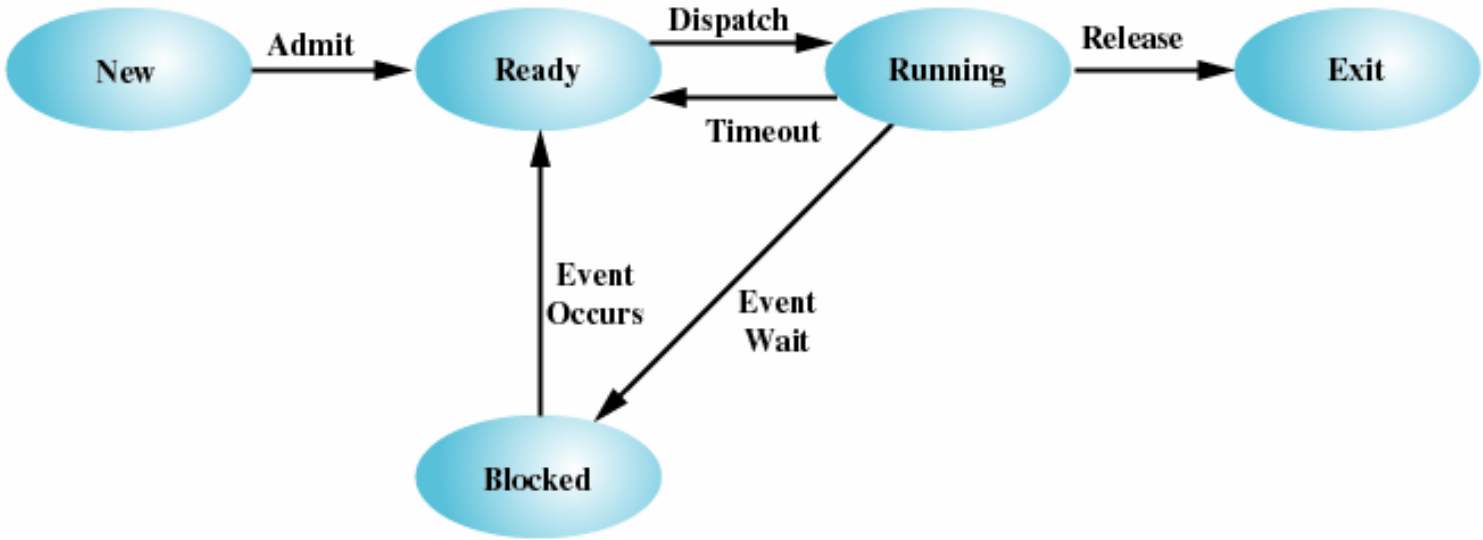
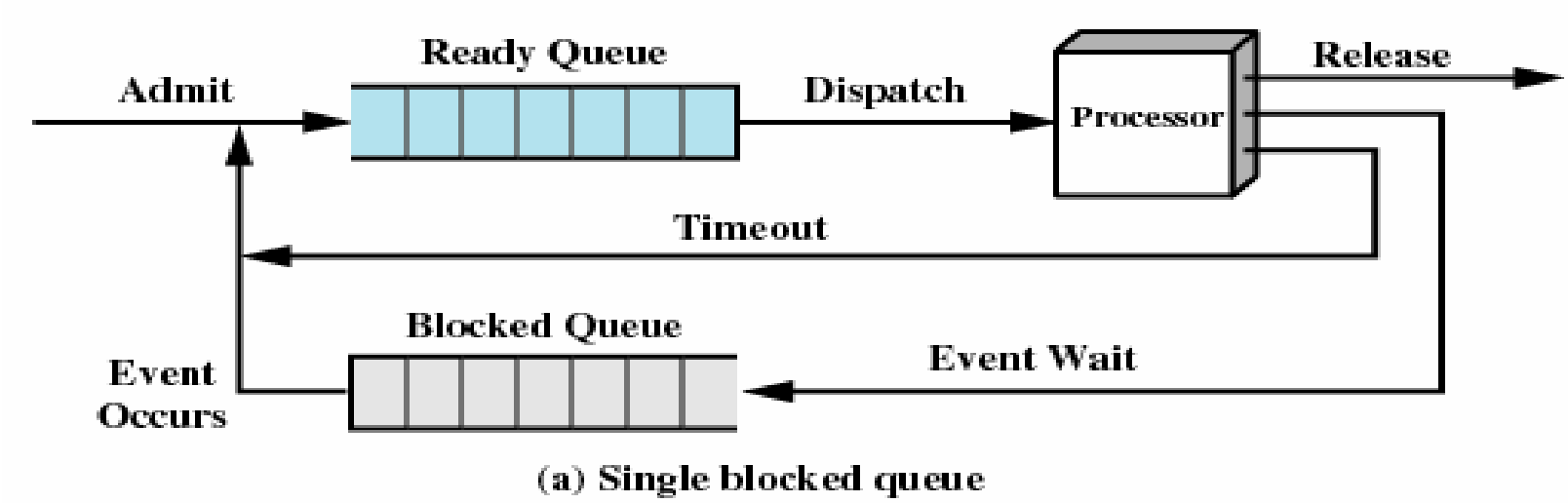


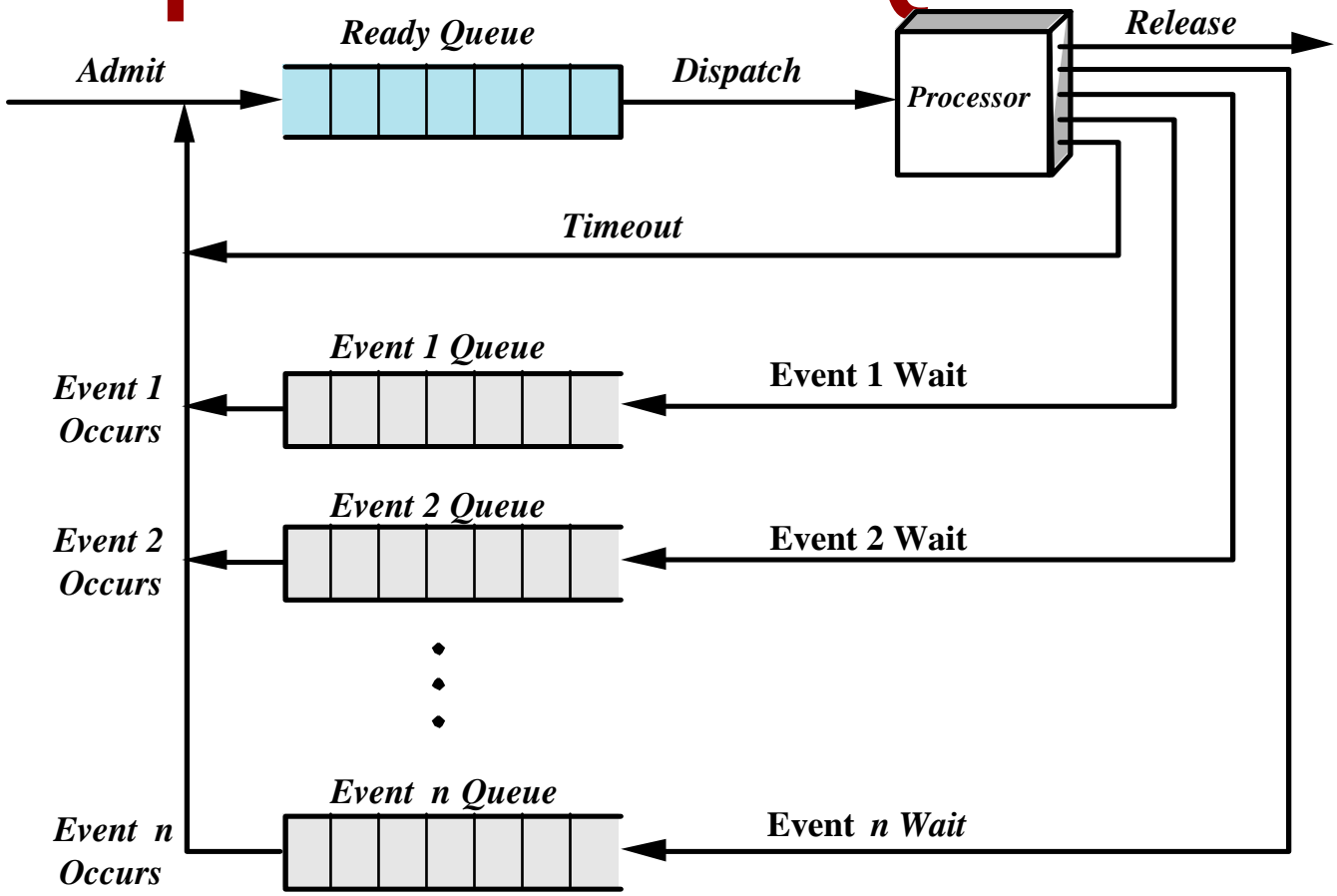
Figure 3.6 Five-State Process Model

Using Two Queues



(a) Single blocked queue

Multiple Blocked Queues



(b) Multiple blocked queues

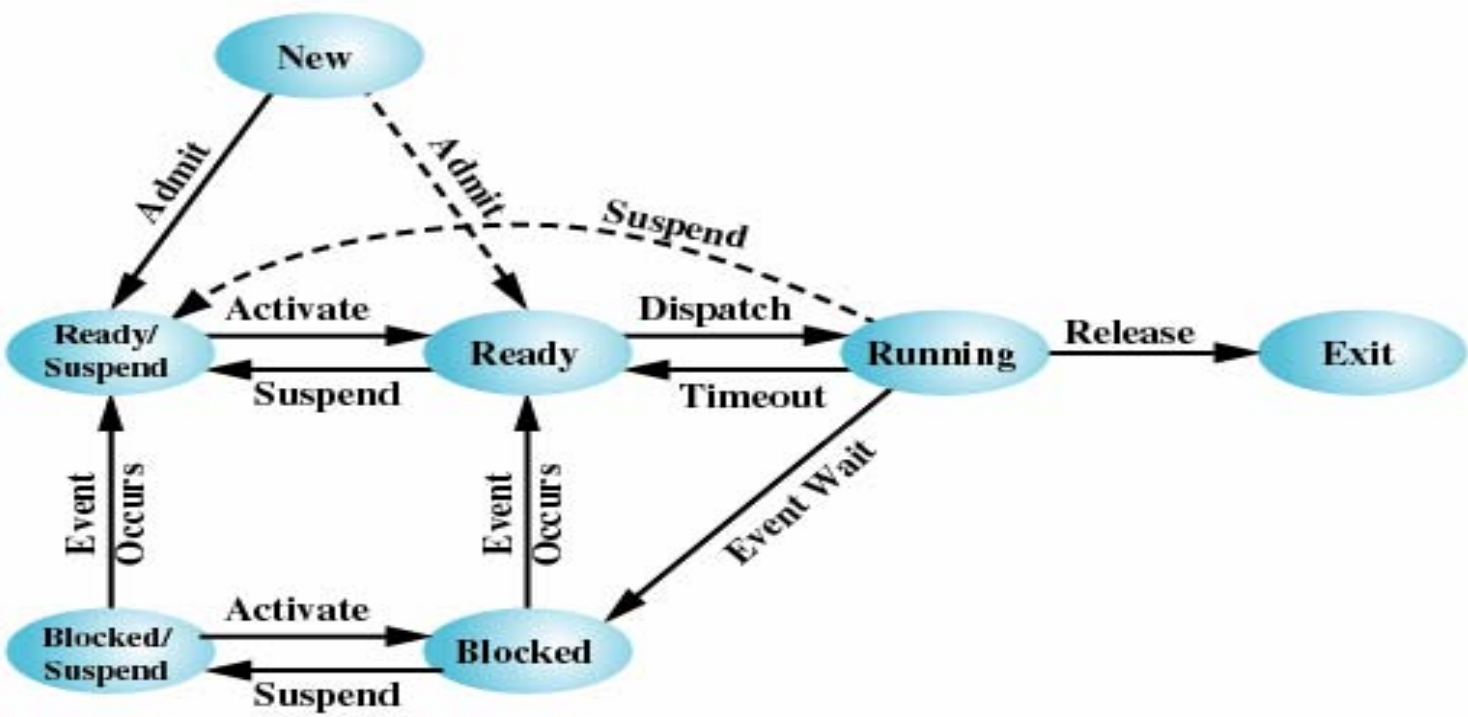
UPPSALA UNIVERSITET

Informationsteknologi

Suspended Processes

- Processor is faster than I/O so all processes could be waiting for I/O
- Swap these processes to disk to free up more memory
- Blocked state becomes suspend state when swapped to disk
- Two new states
  - Blocked/Suspend
  - Ready/Suspend

Two Suspend States



(b) With Two Suspend States

Figure 3.9 Process State Transition Diagram with Suspend States

Reasons for Process Suspension

Table 3.3 Reasons for Process Suspension

Swapping	The operating system needs to release sufficient main memory to bring in a process that is ready to execute.
Other OS reason	The operating system may suspend a background or utility process or a process that is suspected of causing a problem.
Interactive user request	A user may wish to suspend execution of a program for purposes of debugging or in connection with the use of a resource.
Timing	A process may be executed periodically (e.g., an accounting or system monitoring process) and may be suspended while waiting for the next time interval.
Parent process request	A parent process may wish to suspend execution of a descendent to examine or modify the suspended process, or to coordinate the activity of various descendents.

Processes and Resources

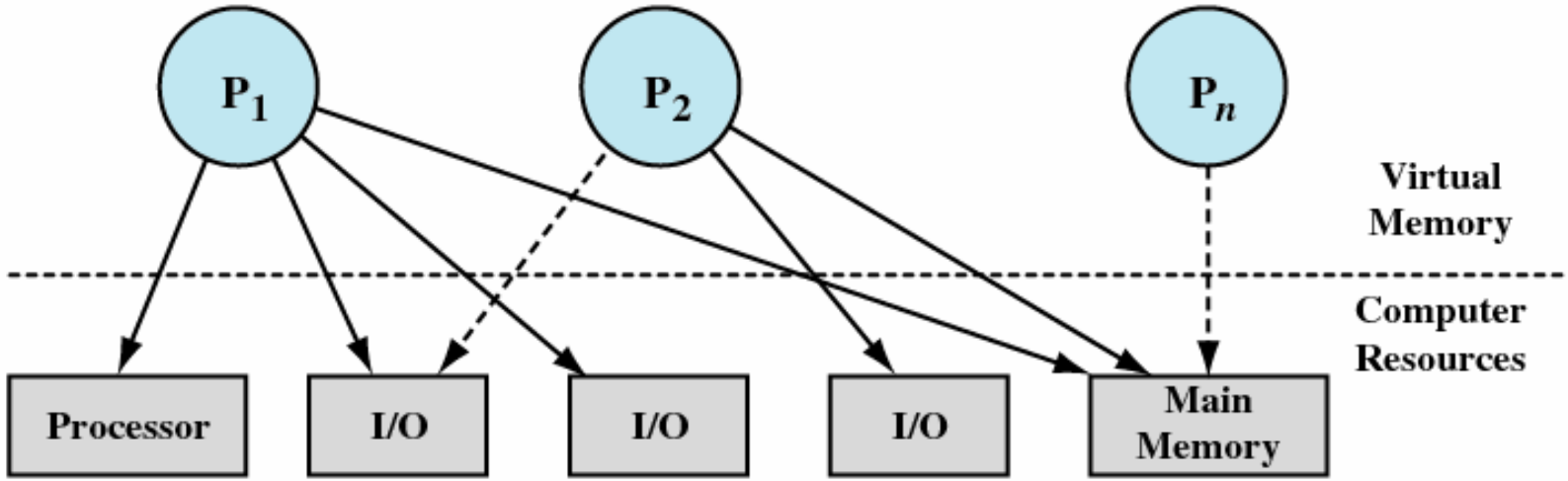


Figure 3.10 Processes and Resources (resource allocation at one snapshot in time)

Operating System Control Structures

- Information about the current status of each process and resource
- Tables are constructed for each entity the operating system manages
  - Memory
  - Devices
  - Files
  - Processes

UPPSALA UNIVERSITET

Informationsteknologi

Memory Tables

- Allocation of main memory to processes
- Allocation of secondary (virtual) memory to processes
- Protection attributes for access to shared memory regions
- Information needed to manage virtual memory

I/O Tables

- I/O device is available or assigned
- Status of I/O operation
- Location in main memory being used as the source or destination of the I/O transfer

File Tables

- Existence of files
- Location on secondary memory
- Current status
- Attributes
- Sometimes this information is maintained by a file management system

Process Table

- Where process is located
- Attributes in the process control block
  - Program
  - Data
  - Stack

Process Image

Table 3.4 Typical Elements of a Process Image

User Data
The modifiable part of the user space. May include program data, a user stack area, and programs that may be modified.

User Program
The program to be executed.

System Stack
Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls.

Process Control Block
Data needed by the operating system to control the process (see Table 3.5).

Process Control Block

- Process identification
  - Identifiers
    - Numeric identifiers that may be stored with the process control block include
      - Identifier of this process
      - Identifier of the process that created this process (parent process)
      - User identifier

UPPSALA UNIVERSITET

Informationsteknologi

Process Control Block

- Processor State Information
  - User-Visible Registers
    - A user-visible register is one that may be referenced by means of the machine language that the processor executes while in user mode. Typically, there are from 8 to 32 of these registers, although some RISC implementations have over 100.

Process Control Block

- Processor State Information
  - Control and Status Registers
    - These are a variety of processor registers that are employed to control the operation of the processor. These include
      - *Program counter:* Contains the address of the next instruction to be fetched
      - *Condition codes:* Result of the most recent arithmetic or logical operation (e.g., sign, zero, carry, equal, overflow)
      - *Status information:* Includes interrupt enabled/disabled flags, execution mode

Process Control Block

- Processor State Information
  - Stack Pointers
    - Each process has one or more last-in-first-out (LIFO) system stacks associated with it. A stack is used to store parameters and calling addresses for procedure and system calls. The stack pointer points to the top of the stack.

UPPSALA UNIVERSITET

Informationsteknologi

Process Control Block

- Process Control Information
  - Scheduling and State Information
    - This is information that is needed by the operating system to perform its scheduling function. Typical items of information:
      - *Process state:* defines the readiness of the process to be scheduled for execution (e.g., running, ready, waiting, halted).
      - *Priority:* One or more fields may be used to describe the scheduling priority of the process. In some systems, several values are required (e.g., default, current, highest-allowable)
      - *Scheduling-related information:* This will depend on the scheduling algorithm used. Examples are the amount of time that the process has been waiting and the amount of time that the process executed the last time it was running.
      - *Event:* Identity of event the process is awaiting before it can be resumed

UPPSALA UNIVERSITET

Informationsteknologi

Process Control Block

- Process Control Information
  - Data Structuring
    - A process may be linked to other process in a queue, ring, or some other structure. For example, all processes in a waiting state for a particular priority level may be linked in a queue. A process may exhibit a parent-child (creator-created) relationship with another process. The process control block may contain pointers to other processes to support these structures.

Process Control Block

- Process Control Information
  - Interprocess Communication
    - Various flags, signals, and messages may be associated with communication between two independent processes. Some or all of this information may be maintained in the process control block.
  - Process Privileges
    - Processes are granted privileges in terms of the memory that may be accessed and the types of instructions that may be executed. In addition, privileges may apply to the use of system utilities and services.

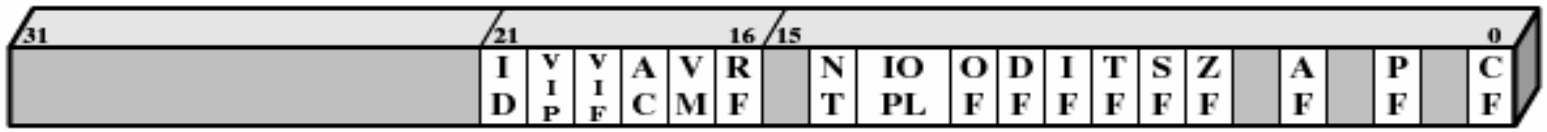
Process Control Block

- Process Control Information
  - Memory Management
    - This section may include pointers to segment and/or page tables that describe the virtual memory assigned to this process.
  - Resource Ownership and Utilization
    - Resources controlled by the process may be indicated, such as opened files. A history of utilization of the processor or other resources may also be included; this information may be needed by the scheduler.

Processor State Information

- Contents of processor registers
  - User-visible registers
  - Control and status registers
  - Stack pointers
- Program status word (PSW)
  - contains status information
  - Example: the EFLAGS register on Pentium machines

Pentium II EFLAGS Register



ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag

DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

Figure 3.12 Pentium II EFLAGS Register

Modes of Execution

- User mode
  - Less-privileged mode
  - User programs typically execute in this mode
- System mode, control mode, or kernel mode
  - More-privileged mode
  - Kernel of the operating system

UPPSALA UNIVERSITET

Informationsteknologi

Process Creation

- Assign a unique process identifier
- Allocate space for the process
- Initialize process control block
- Set up appropriate linkages
  - Ex: add new process to linked list used for scheduling queue
- Create of expand other data structures
  - Ex: maintain an accounting file

When to Switch a Process

- Clock interrupt
  - process has executed for the maximum allowable time slice
- I/O interrupt
- Memory fault
  - memory address is in virtual memory so it must be brought into main memory

When to Switch a Process

- Trap
  - error or exception occurred
  - may cause process to be moved to Exit state
- Supervisor call
  - such as file open

UPPSALA UNIVERSITET

Informationsteknologi

Change of Process State

- Save context of processor including program counter and other registers
- Update the process control block of the process that is currently in the running state
- Move process control block to appropriate queue – ready; blocked; ready/suspend
- Select another process for execution

UPPSALA UNIVERSITET

Informationsteknologi

Change of Process State

- Update the process control block of the process selected
- Update memory-management data structures
- Restore context of the selected process

UNIX Process States

Table 3.9 UNIX Process States

User Running	Executing in user mode.
Kernel Running	Executing in kernel mode.
Ready to Run, in Memory	Ready to run as soon as the kernel schedules it.
Asleep in Memory	Unable to execute until an event occurs; process is in main memory (a blocked state).
Ready to Run, Swapped	Process is ready to run, but the swapper must swap the process into main memory before the kernel can schedule it to execute.
Sleeping, Swapped	The process is awaiting an event and has been swapped to secondary storage (a blocked state).
Preempted	Process is returning from kernel to user mode, but the kernel preempts it and does a process switch to schedule another process.
Created	Process is newly created and not yet ready to run.
Zombie	Process no longer exists, but it leaves a record for its parent process to collect.

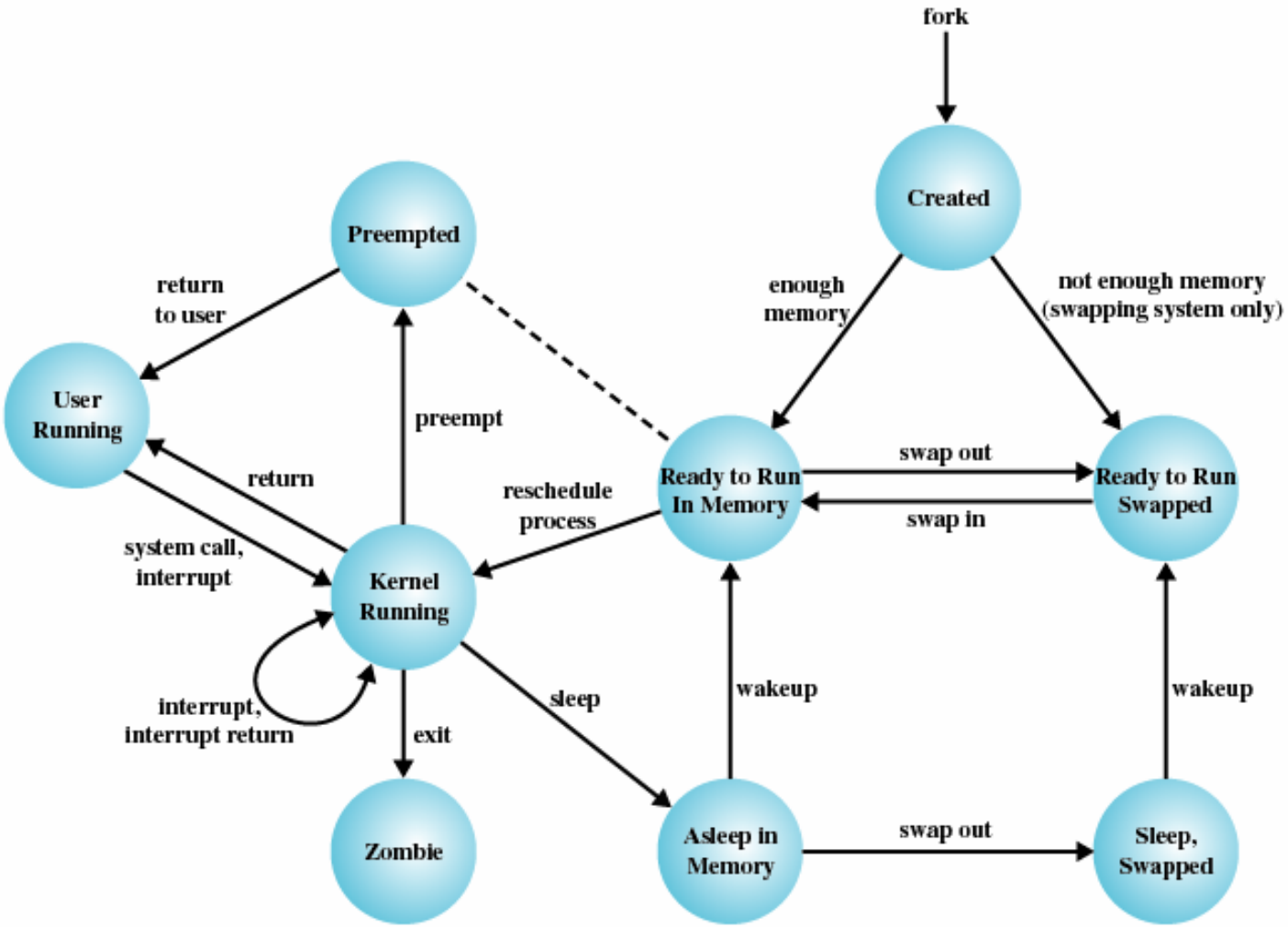


Figure 3.17 UNIX Process State Transition Diagram