Sistemas Operativos

Concepto y planificación de procesos

Departamento de Ingeniería en Sistemas y Computación
Universidad Católica del Norte, Antofagasta.

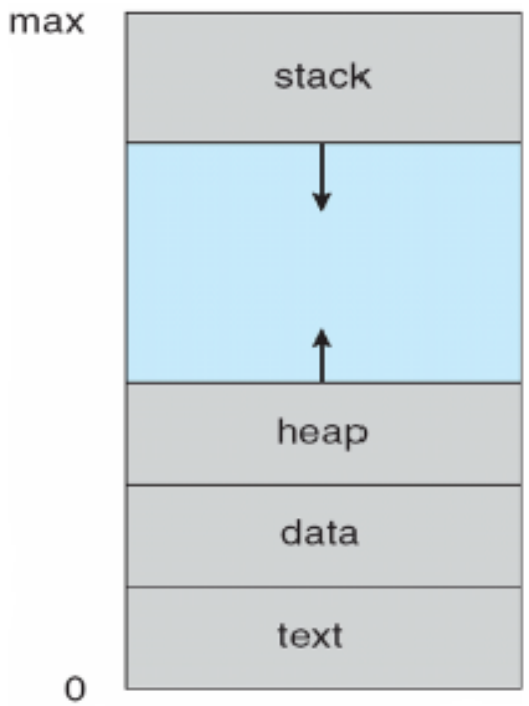
Procesos

- Un proceso es un programa en ejecución.
- La ejecución de un proceso es secuencial.
- Todo proceso necesita
  - PC : program counter
  - Stack
  - Región de datos

Procesos

- Un proceso es un programa en ejecución.
- La ejecución de un proceso es secuencial.
- Todo proceso necesita
  - PC : program counter
  - Stack
  - Región de datos

Un proceso en memoria



Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

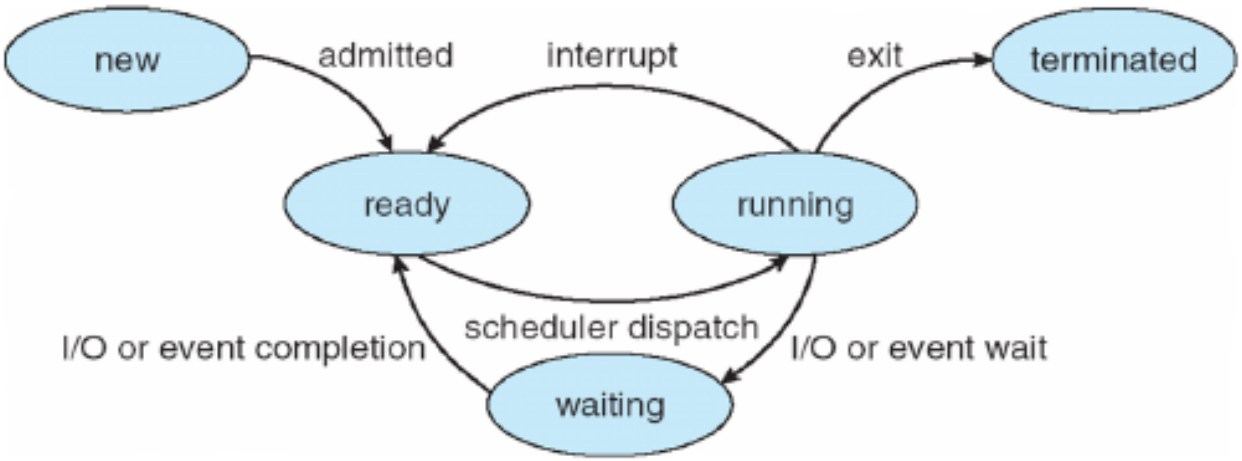
- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Estados de proceso

En la medida que un proceso se ejecuta, va cambiando de estado. Estos estados son:

- new: el proceso es creado
- running: las instrucciones están siendo ejecutadas
- waiting: el proceso espera que ocurra algún evento
- ready: el proceso espera que se le asigne un procesador
- terminated: el proceso terminó su ejecución

Diagrama de estados



Process Control Block

La estructura de datos que mantiene la información asociada a cada proceso, se denomina PCB. Contiene:

- Estado del proceso
- Program counter
- Registros de CPU
- Información de itineración de CPU
- Información de gestión de memoria
- Información de estado de E/S

Process Control Block

La estructura de datos que mantiene la información asociada a cada proceso, se denomina PCB. Contiene:

- Estado del proceso
- Program counter
- Registros de CPU
- Información de itineración de CPU
- Información de gestión de memoria
- Información de estado de E/S

Process Control Block

La estructura de datos que mantiene la información asociada a cada proceso, se denomina PCB. Contiene:

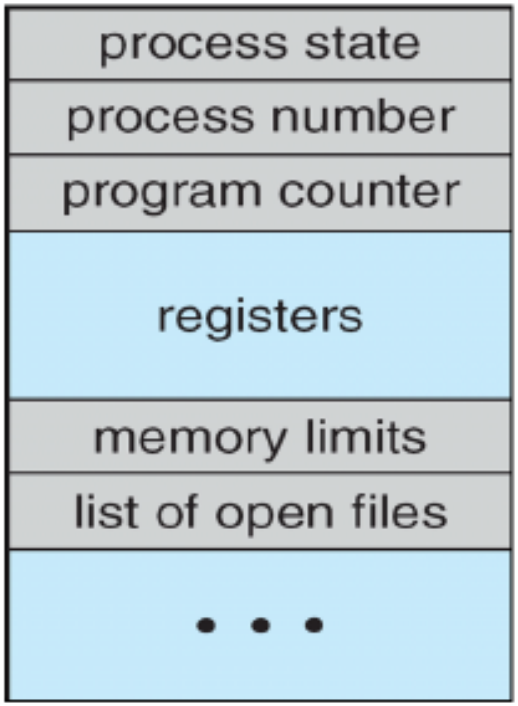
- Estado del proceso
- Program counter
- Registros de CPU
- Información de itineración de CPU
- Información de gestión de memoria
- Información de estado de E/S

Process Control Block

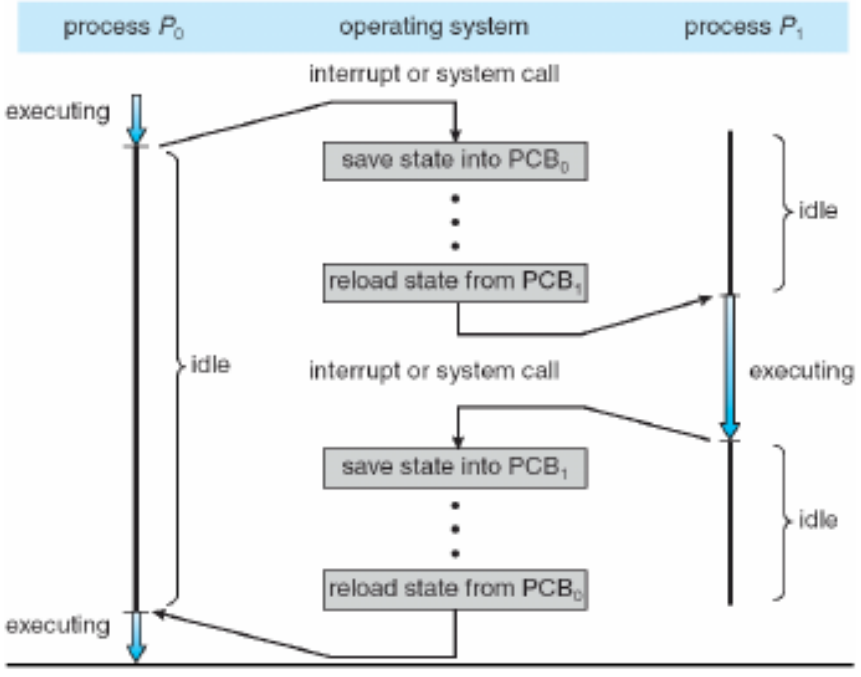
La estructura de datos que mantiene la información asociada a cada proceso, se denomina PCB. Contiene:

- Estado del proceso
- Program counter
- Registros de CPU
- Información de itineración de CPU
- Información de gestión de memoria
- Información de estado de E/S

Process Control Block



Conmutación entre procesos



Itineración de procesos

El SO utiliza las siguientes colas para la itineración de procesos:

- Cola de jobs: conjunto de todos los procesos del sistema
- Cola ready: conjunto de todos los procesos que están en memoria principal listos y esperando ser ejecutados
- Cola de dispositivos: procesos que esperan por E/S

Los procesos se mueven entre estas colas en su ciclo de vida.

Itineración de procesos

El SO utiliza las siguientes colas para la itineración de procesos:

- Cola de jobs: conjunto de todos los procesos del sistema
- Cola ready: conjunto de todos los procesos que están en memoria principal listos y esperando ser ejecutados
- Cola de dispositivos: procesos que esperan por E/S

Los procesos se mueven entre estas colas en su ciclo de vida.

Itineración de procesos

El SO utiliza las siguientes colas para la itineración de procesos:

- Cola de jobs: conjunto de todos los procesos del sistema
- Cola ready: conjunto de todos los procesos que están en memoria principal listos y esperando ser ejecutados
- Cola de dispositivos: procesos que esperan por E/S

Los procesos se mueven entre estas colas en su ciclo de vida.

Itineración de procesos

El SO utiliza las siguientes colas para la itineración de procesos:

- Cola de jobs: conjunto de todos los procesos del sistema
- Cola ready: conjunto de todos los procesos que están en memoria principal listos y esperando ser ejecutados
- Cola de dispositivos: procesos que esperan por E/S

Los procesos se mueven entre estas colas en su ciclo de vida.

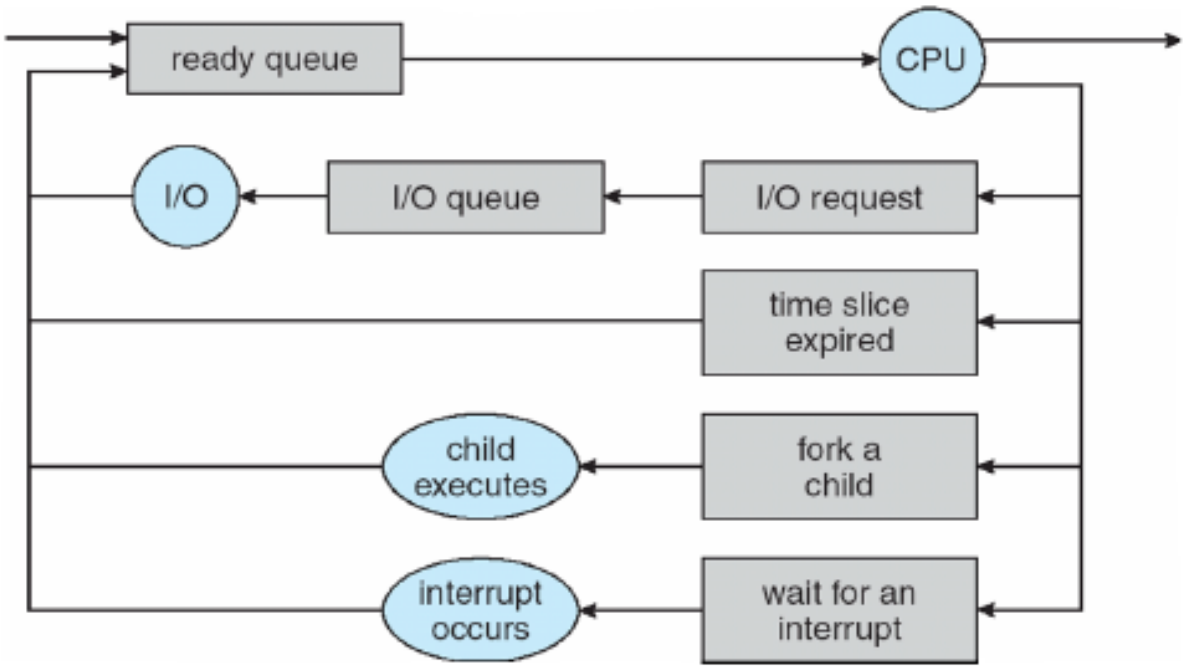
Itineración de procesos

El SO utiliza las siguientes colas para la itineración de procesos:

- Cola de jobs: conjunto de todos los procesos del sistema
- Cola ready: conjunto de todos los procesos que están en memoria principal listos y esperando ser ejecutados
- Cola de dispositivos: procesos que esperan por E/S

Los procesos se mueven entre estas colas en su ciclo de vida.

Itineración de procesos



Itineradores

Existen dos tipos:

- De largo plazo (job scheduler): selecciona cual proceso se debe llevar a la cola ready. Puede que en algunos SO este scheduler no exista.
- De corto plazo (CPU scheduler): selecciona el proceso a ejecutar a continuación y asigna CPU.

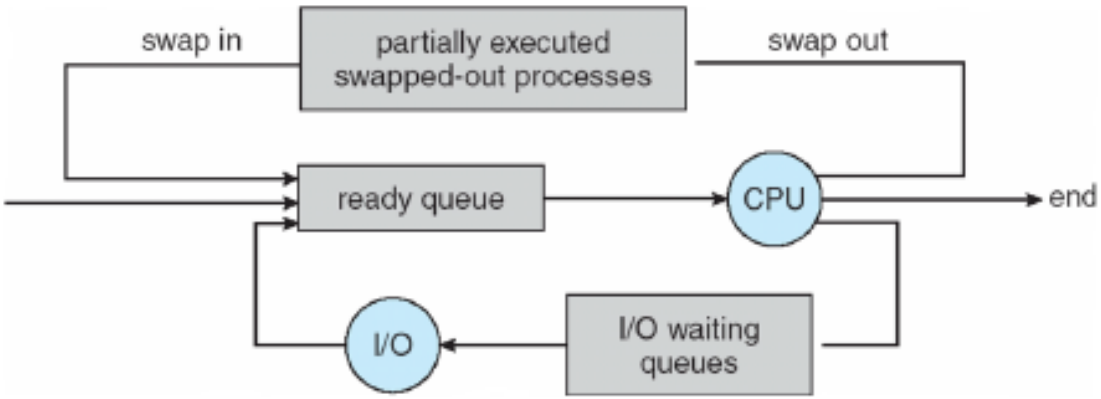
Itineradores

Existen dos tipos:

- De largo plazo (job scheduler): selecciona cual proceso se debe llevar a la cola ready. Puede que en algunos SO este scheduler no exista.
- De corto plazo (CPU scheduler): selecciona el proceso a ejecutar a continuación y asigna CPU.

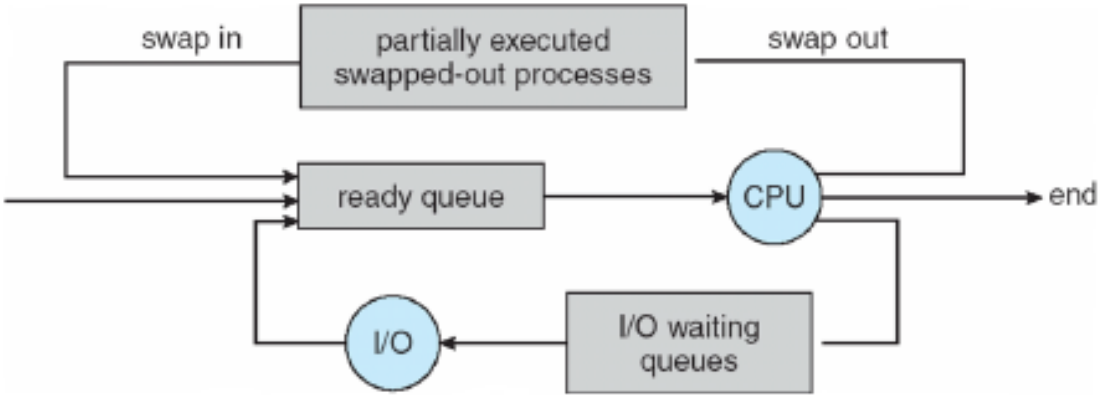
Itinerador de mediano plazo

- Este esquema se denomina swapping
- Reduce el grado de multiprogramación (jobs en memoria principal)



Itinerador de mediano plazo

- Este esquema se denomina swapping
- Reduce el grado de multiprogramación (jobs en memoria principal)



Itineradores

- El itinerador de corto plazo es invocado frecuentemente (orden de milisegundos). Debe ser rápido.
- El itinerador de largo plazo se invoca en forma poco frecuente (orden de segundos o minutos). Puede ser lento
- El itinerador de mediano plazo controla el grado de multiprogramación
- Profundizaremos en planificación de procesos en la Unidad III

Itineradores

- El itinerador de corto plazo es invocado frecuentemente (orden de milisegundos). Debe ser rápido.
- El itinerador de largo plazo se invoca en forma poco frecuente (orden de segundos o minutos). Puede ser lento
- El itinerador de mediano plazo controla el grado de multiprogramación
- Profundizaremos en planificación de procesos en la Unidad III

Itineradores

- El itinerador de corto plazo es invocado frecuentemente (orden de milisegundos). Debe ser rápido.
- El itinerador de largo plazo se invoca en forma poco frecuente (orden de segundos o minutos). Puede ser lento
- El itinerador de mediano plazo controla el grado de multiprogramación
- Profundizaremos en planificación de procesos en la Unidad III

Itineradores

- El itinerador de corto plazo es invocado frecuentemente (orden de milisegundos). Debe ser rápido.
- El itinerador de largo plazo se invoca en forma poco frecuente (orden de segundos o minutos). Puede ser lento
- El itinerador de mediano plazo controla el grado de multiprogramación
- Profundizaremos en planificación de procesos en la Unidad III

Tipos de procesos

Los procesos se pueden clasificar en dos tipos:

- Limitados por E/S: pasan más tiempo haciendo E/S que cómputos. Muchas ráfagas cortas de CPU.
- Limitados por CPU: pasan más tiempo haciendo cómputos. Pocas ráfagas largas de CPU.

Tipos de procesos

Los procesos se pueden clasificar en dos tipos:

- Limitados por E/S: pasan más tiempo haciendo E/S que cómputos. Muchas ráfagas cortas de CPU.
- Limitados por CPU: pasan más tiempo haciendo cómputos. Pocas ráfagas largas de CPU.

Tipos de procesos

Los procesos se pueden clasificar en dos tipos:

- Limitados por E/S: pasan más tiempo haciendo E/S que cómputos. Muchas ráfagas cortas de CPU.
- Limitados por CPU: pasan más tiempo haciendo cómputos. Pocas ráfagas largas de CPU.

Operaciones sobre procesos

- El SO debe proveer un mecanismo para crear y terminar un proceso
- La mayoría de los SO permiten la ejecución (pseudo) concurrente y pueden ser creados y eliminados dinámicamente

Operaciones sobre procesos

- El SO debe proveer un mecanismo para crear y terminar un proceso
- La mayoría de los SO permiten la ejecución (pseudo) concurrente y pueden ser creados y eliminados dinámicamente

Creación de procesos

- Cada proceso tiene un identificador que se denomina **pid**
- El pid permite realizar diversas operaciones sobre procesos
- Un proceso padre puede crear procesos hijos, los cuales a su vez pueden crear otros procesos creando un árbol de procesos.

Creación de procesos

- Cada proceso tiene un identificador que se denomina **pid**
- El pid permite realizar diversas operaciones sobre procesos
- Un proceso padre puede crear procesos hijos, los cuales a su vez pueden crear otros procesos creando un árbol de procesos.

Creación de procesos

- Cada proceso tiene un identificador que se denomina **pid**
- El pid permite realizar diversas operaciones sobre procesos
- Un proceso padre puede crear procesos hijos, los cuales a su vez pueden crear otros procesos creando un árbol de procesos.

Creación de procesos

Existen tres opciones para la compartición de recursos:

- Padres e hijos comparten todos sus recursos
- Los hijos comparten un subconjunto de los recursos del padre
- Padres e hijos no comparten recursos

Creación de procesos

Existen tres opciones para la compartición de recursos:

- Padres e hijos comparten todos sus recursos
- Los hijos comparten un subconjunto de los recursos del padre
- Padres e hijos no comparten recursos

Creación de procesos

Existen tres opciones para la compartición de recursos:

- Padres e hijos comparten todos sus recursos
- Los hijos comparten un subconjunto de los recursos del padre
- Padres e hijos no comparten recursos

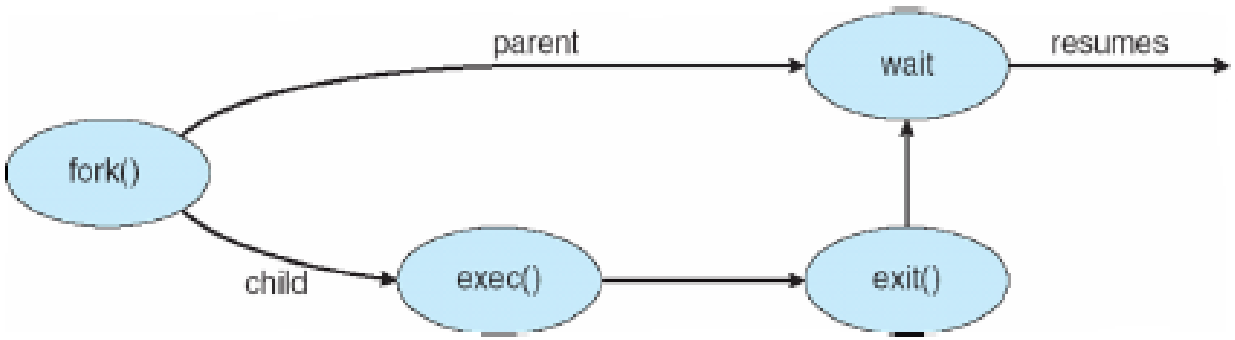
Ejecución de procesos

Dos alternativas para la ejecución:

- Padres e hijos se ejecutan concurrentemente
- El padre espera hasta que el hijo termine

Ejemplo UNIX

La llamada al sistema **fork** crea un nuevo proceso



Ejemplo en C

```c
#include <stdlib.h>>
#include <unistd.h>
#include <stdio.h>
int main()
{
int  pid;
        /* fork another process */
        pid = fork();
        if (pid < 0) { /* error occurred */
                fprintf(stderr, "Fork Failed");
                exit(-1);
        }
        else if (pid == 0) { /* child process */
                execlp("/bin/date", "date", NULL);
        }
        else { /* parent process */
                /* parent will wait for the child to complete */
                wait (NULL);
                printf ("Child Complete\n");
                exit(0);
        }
}
```

Planificación en multiprocesadores

- Multiprocesador: máquina con conjunto de procesadores que comparten un mismo espacio de direcciones de memoria física.
- Aplicaciones modernas están constituidas por `threads`, que se comunican por memoria compartida.
- Se debe reconocer el thread como entidad planificable.

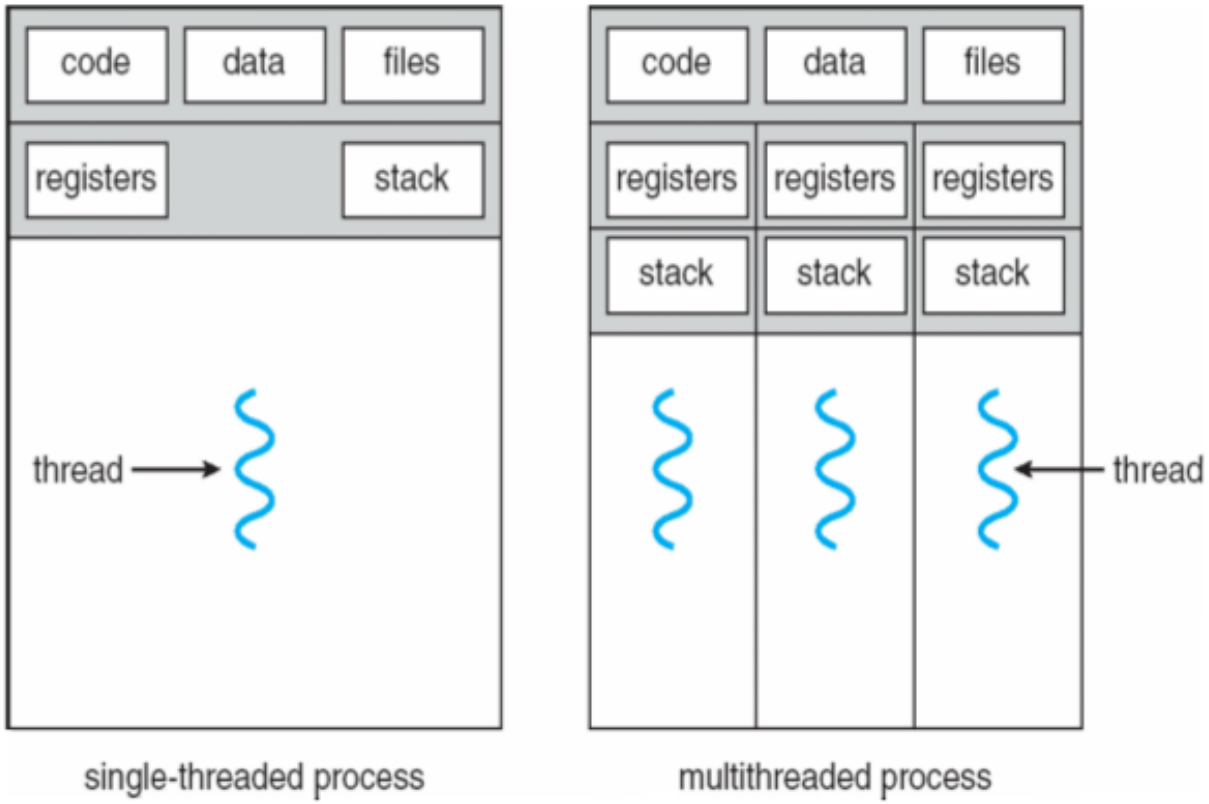
Planificación en multiprocesadores

- Multiprocesador: máquina con conjunto de procesadores que comparten un mismo espacio de direcciones de memoria física.
- Aplicaciones modernas están constituidas por `threads`, que se comunican por memoria compartida.
- Se debe reconocer el thread como entidad planificable.

Planificación en multiprocesadores

- Multiprocesador: máquina con conjunto de procesadores que comparten un mismo espacio de direcciones de memoria física.
- Aplicaciones modernas están constituidas por `threads`, que se comunican por memoria compartida.
- Se debe reconocer el thread como entidad planificable.

Planificación en multiprocesadores



Threads

Beneficios:

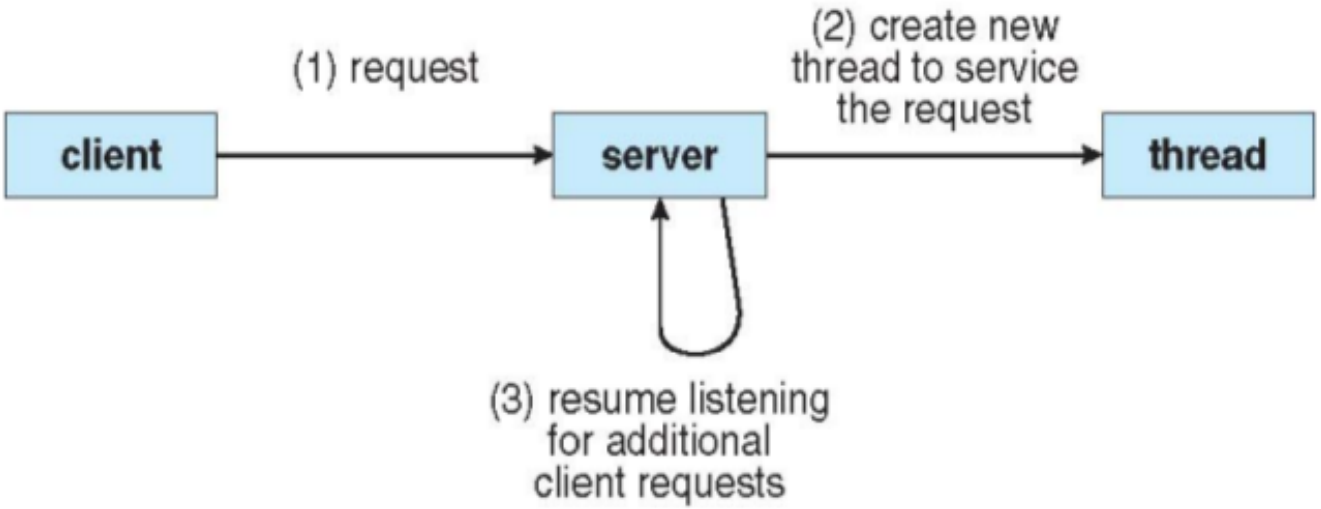
- Sensibilidad: Permite que un programa continue corriendo aún si parte de él está bloqueado o realizando una operación lenta
- Compartición de recursos: Comparten memoria y recursos que el proceso tiene por defecto

Threads

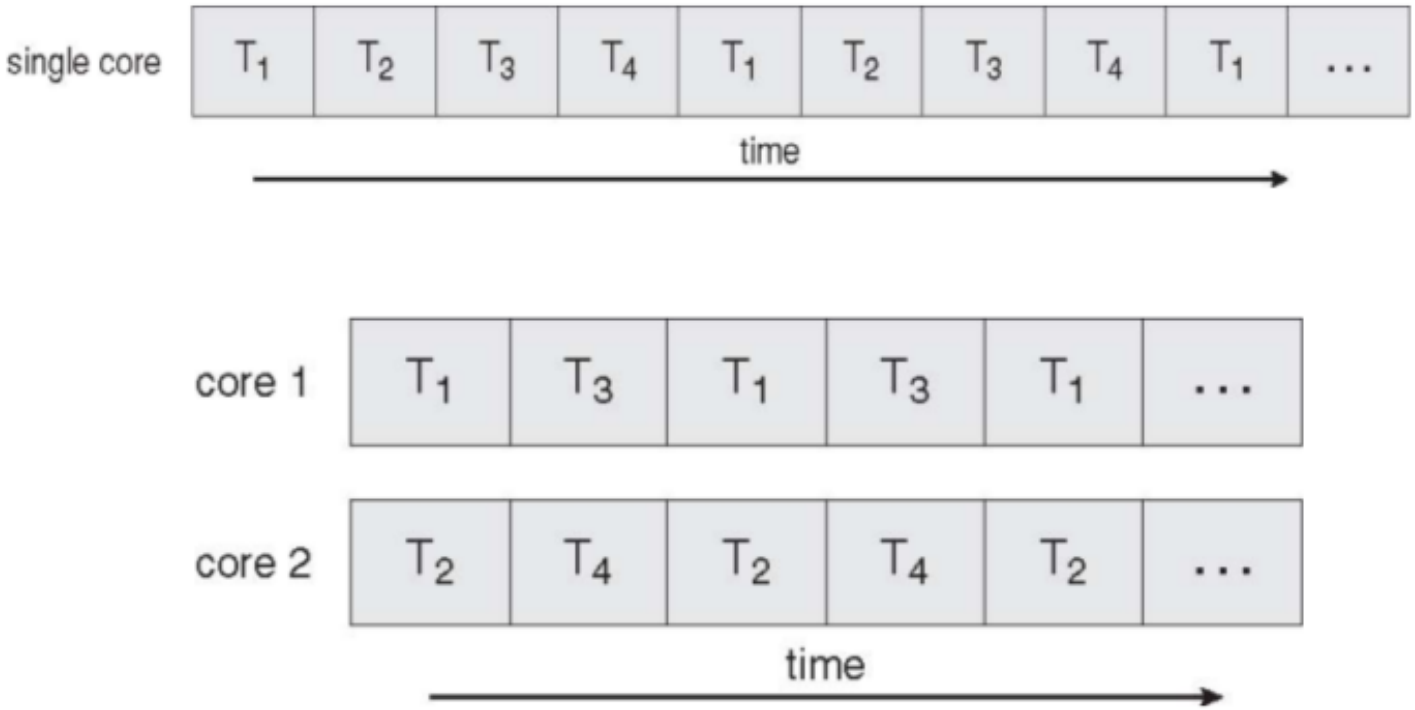
Beneficios:

- Sensibilidad: Permite que un programa continue corriendo aún si parte de él está bloqueado o realizando una operación lenta
- Compartición de recursos: Comparten memoria y recursos que el proceso tiene por defecto

Servidor Multithread



Ejecución concurrente vs paralela



Threads de usuarios

Principales bibliotecas:

- POSIX Pthreads
- Win32 threads
- Java threads

Pthreads

- Estándar POSIX para threads (IEEE 1003.1c)
- Se utiliza en SO UNIX

Pthreads

- Estándar POSIX para threads (IEEE 1003.1c)
- Se utiliza en SO UNIX

Mostrar el *i*-ésimo número primo

```
/*primos.c en ... Sources/Ch4*/
/*Calcula los numeros primos de 2 hasta el numero N
En este ejemplo, N=5000 y es pasado a  *ARG.
se muestra el primo mayor encontrado */
void* compute_prime (void* arg)
{
        int candidate = 2;
        int n = *((int*) arg);
        while (1) {
                int factor;
                int is_prime = 1;
                /* Test primality by successive division. */
                for (factor = 2; factor < candidate; ++factor)
                        if (candidate % factor == 0) {
                                is_prime = 0;
                                break;
                        }
                /* Es el primo que buscamos? */
                if (is_prime) {
                        if (--n == 0)
                        /* Return the desired prime number as the thread return value.*/
                                return (void*) candidate;
                }
                ++candidate;
        }
        return NULL;
}
```

Mostrar el *i*-ésimo número primo

```c
int main ()
{
        pthread_t thread;
        pthread_attr_t attr;
        int which_prime =5000;
        int prime;
        pthread_attr_init(&attr);
        /* Start the computing thread, up to the 5,000th prime number. */
        pthread_create (&thread, &attr, &compute_prime, &which_prime);
        /* Do some other work here... */
        /* Wait for the prime number thread to complete, and get the result.*/
        pthread_join (thread, (void*) &prime);
        /* Print the largest prime it computed. */
        printf("The %dth prime number is %d.\n", which_prime, prime);
        return 0;
}
```

Threads en Python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#durmiendo.py   /Sources/Ch4/Python
import time
from threading import Thread

def myfunc(i):
    print "durmiendo 5 segundos desde el thread %d" % i
    time.sleep(5)
    print "despertando desde el thread %d" % i

for i in range(10):
    t = Thread(target=myfunc, args=(i,))
    t.start()
```

Threads en Python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import threading

theVar = 1

class MyThread ( threading.Thread ):

    def run ( self ):
        global theVar
        print 'Soy el thread numero ' + str ( theVar ) + ' activo.'
        print 'Hola y Chao.'
        theVar = theVar + 1

for x in xrange ( 7 ):
    MyThread().start()
```