Centro de
Investigación en
Computación

Process Management

Administrador de Procesos

Teoría general

Linux

Windows

Course

Operating System (with focus on Security)

Instructor

Acosta Bermejo Raúl

Lecture notes

2024-B
28 de agosto del 2024
Última actualizacioón

Instituto
Politécnico
Nacional

Table of contents (outline)

Tabla de contenido

1. Introducción
2. Planificadores
    1. Teoría general
    2. Algoritmos de planificación
3. Casos de Uso
    1. Dos/Windows
    2. Linux
4. Temas avanzados
5. Referencias

Introduction

Conceptos básicos

Process management

- The OS must:
  - Allocate resources to processes.
  - Enable processes to share and exchange information.
  - Protect the resources of each process from other processes.
  - Enable synchronisation among processes.
- To meet these requirements, the OS must maintain a data structure for each process, which describes the state and resource ownership of that process, and which enables the OS to exert control over each process.

El objetivo es permitir que varios usuarios utilicen el mismo recurso al mismo tiempo (procesador(es), multi-tasking)

Introduction

Conceptos básicos

Proceso: es un programa (sólo el código) en ejecución.

- Significa que está en RAM y que
- Hay ciertos valores en los registros del procesador, y
- Lo mismo para el modelo de funciones (valores en la pila).

Segmentos Data Pointer (DP)	Datos (Data) Variables globales Heap
Instruction Pointer (IP)	Código (Text) Funciones
Stack Pointer (SP)	Pila (Stack) Variables locales Parámetros

Base Pointer
Specific Pointer

Introduction

Conceptos básicos

Scheduler: es el algoritmo que decide cual de los N procesos se ejecuta.

- Planificador en español.
- Existen varios tipos de algoritmos con su TDA.
- TDA (Tipo de Dato Abstracto): normalmente una cola (s) y una tabla.
- Tipo de planificador: para Tiempo Real (RT),
  para el Espacio de usuario o kernel.

Quantum: es intervalo de tiempo en que se ejecuta un proceso.

- Varía con cada S.O. (velocidad del procesador) **Time-sharing**.

Introduction

Conceptos básicos

Cambio de contexto:

- Es el algoritmo que realiza físicamente el intercambio de procesos.
- También le llaman dispatcher.
- Depende del Hw y por eso se implementa en ensamblador.

Árbol de procesos: Modelo de control de procesos.

- Define una jerarquía para definir comportamientos.
- Por ej. un proceso padre puede matar al hijo pero no viceversa.
- Manejo de señales: se difunden en una rama del árbol.

Process Identification (PID)

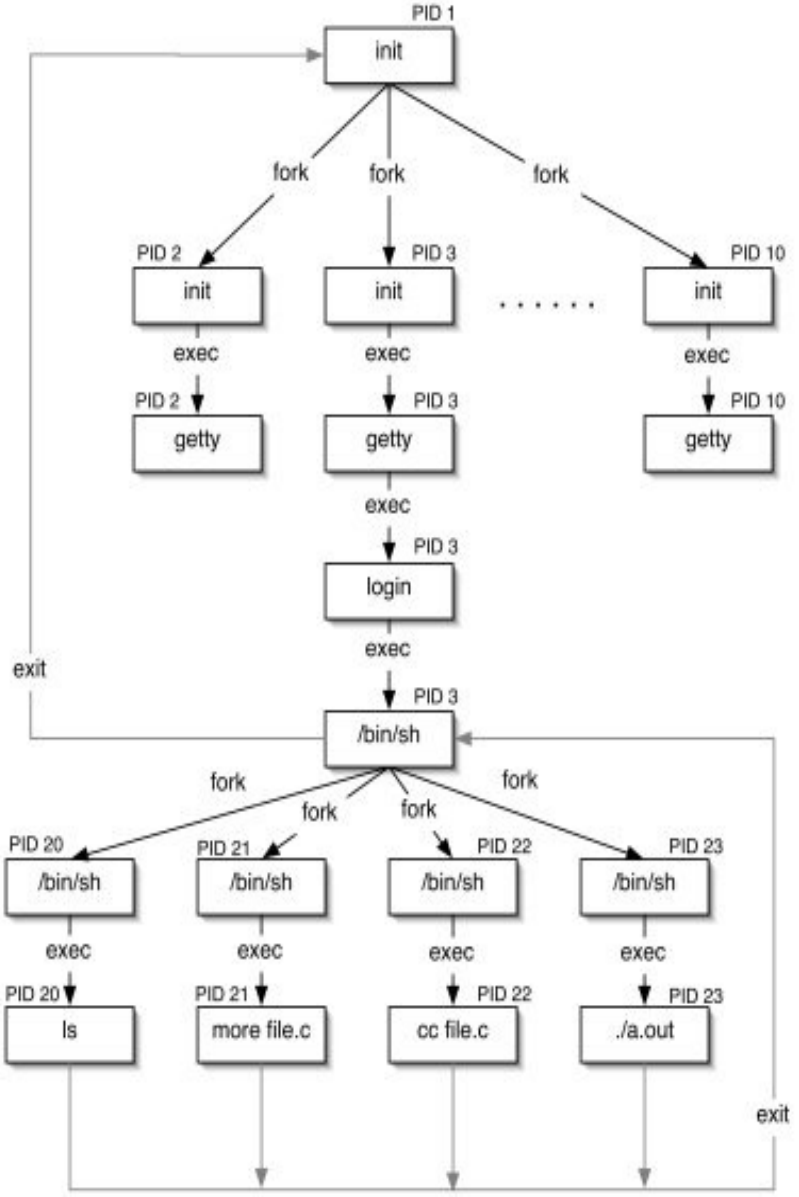
- Identificador de proceso (único y dependiente del S.O.)

Introduction

Conceptos básicos

En Linux hay comandos para ver:

- Tabla de procesos
  - Forma estática
  - Tiempo real
- Árbol de procesos



Introduction

Tabla de procesos

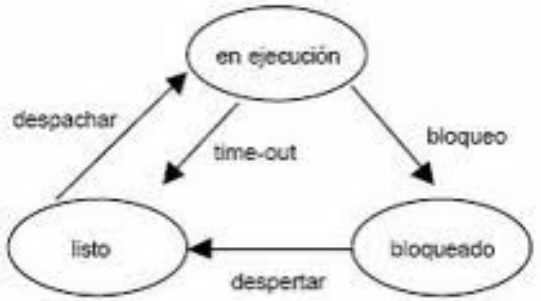
Cada entrada está formado en realidad por el PCB:

- ***Process Control Block***, Bloque de Control de Proceso.
- Esta estructura mantiene apuntadores a su vez a otras estructuras.
- El PCB tiene:
  - Prioridad
  - Zona de memoria
  - Ubicación del ejecutable en disco
  - Dueño del proceso
  - Permisos (que puede hacer, diferentes de archivos)
  - Ambiente de ejecución: variables globales del sistema (shell)
  - Tablas de recursos: archivos abiertos, sockets, semáforos, etc.
  - Etc....

Introduction

Estados de un proceso

- Cada sistema operativo tiene su propios estados reales.
- Para propósitos de estudio se tiene un **modelo**.
- Los estados mínimos son:
  - En ejecución (*running*): el procesador ejecuta las instrucciones del proceso.
  - Listo (*ready*): Espera a que el procesador le de el control.
  - Bloqueado (waiting): Espera a que se libere un recurso (impresora) para poderse ejecutar.
  - Otros: terminado (*terminated*)
- Representación en forma de grafo sencilla.

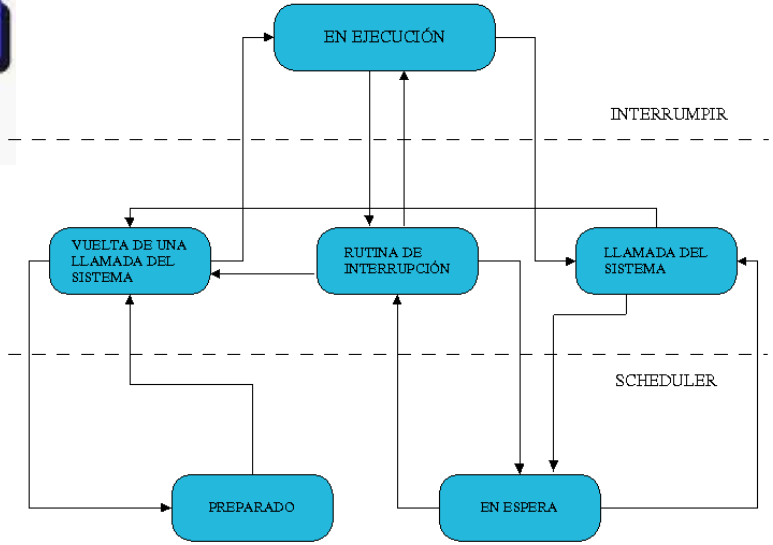


Introduction

Estados de un proceso

Grafos

Representaciones más realistas.



Introduction

Interacción con el sistema

Formas

- API, *Application Programming Interface*
- CLI, *Command-Line Interface*
- GUI, *Graphical User Interface*

Operaciones

- Crear un proceso: shell (CLI), fork (API, clonación)
- Matar un proceso: kill (CLI, API, usa señales).
- Suspender un proceso.
- Mandar una señal.

Introduction

Temas avanzados

Algunos de los temas avanzados son:

- Operaciones remotas (pbm. seguridad)
- Migrar procesos (agentes).
- Procesos distribuidos (SO distribuidos)
  - Planificador de procesos/procesadores distribuido
  - Balanceo de carga.

Schedulers

Planificadores

Teoría

Schedulers

Criterios de optimización

Cada planificador debe ser evaluado en base a la forma en que optimizan las siguientes variables:

1. Maximizar la utilización del CPU.
2. Maximizar el throughput (cantidad de trabajo o información – enviada o almacenada).
3. Minimizar el tiempo de intercambio (turnaround).
4. Minimizar el tiempo de espera.
5. Minimizar el tiempo de respuesta.

Schedulers

Tipos de planificadores

1. Cooperativos o No-Preemptivos.
   - Hasta Win 3.1 y MacOS 9 eran cooperativos.
2. Pre-emptivos
   - A partir de Win NT, y MacOS X.
   - En Linux el cambio fuerte fue usar un Kernel pre-emptivo (ver. 2.4 -> 2.5)
3. Hibridos
   - Windows 95, 98, Me (16 bits/coop, 32 bits/pre)

Schedulers

Tipos de planificadores

1. Procesos limitados por E/S (bound)
   - Pasan la mayor parte del tiempo haciendo operaciones de Entrada/Salida.
   - Cortos tiempos de CPU (bursts).

   Batch, Interactivos, Tiempo Real

2. Procesos limitados por CPU
   - Pasan la mayor parte del tiempo haciendo cálculos.
   - Largos tiempos de CPU.

Schedulers

Tipos de planificadores

Usos de planificadores en el área de computación:

1. Sistema Operativos
2. Redes de computadoras (paquetes, enrutadores, QoS)
3. Clusters/Grids

En otras áreas se presentan como Problemas de optimización:

1. Estocástico (aleatorio o no-determinista) vs determinista.
2. Dinámico vs Estático.

Schedulers

Colas de procesos

Algunas de las principales:

1. Cola de procesos listos (ready)
   Listos para ser ejecutados.
2. Cola de procesos de dispositivos
   Esperan a que se desocupe un dispositivo de E/S.
3. Cola de procesos esperando.
   Esperan a que se de un evento (semáforos, condiciones).

Los procesos son movidos de una cola a otra.

En sistema UNIX-like hay muchas **Estructuras de Datos**:
procesos/hilos
semáforos
archivos
...

Schedulers

Algoritmos de planificación

1. Cola,
   FCFS (*First Come, First Served*), FIFO(*First Input First Ouput*)
2. El trabajo más corto se planifica primero
   SJF (*Shortest Job First*)
3. Planificación por fracciones de tiempo
   RR (*Round-Robin*)
4. Planificación por colas de niveles múltiples
   MLQ (*Multi Level Queues*)
5. Planificación por colas de niveles múltiples con realimentación
   MLFQ (*Multi Level Feedback Queues*)
6. Planificación primero con el tiempo de ejecución más corto
   SRTF (*Shortest Run-Time First*)
7. Planificador de 2 niveles
   Two Level Scheduling.

Schedulers

1. Algoritmo FCFS (First Come, First Served)

Funcionamiento resumido

- Es el algoritmo de planificación más sencillo, basado en una Cola.
- La cola es procesada de la forma más simple: en el orden en que llegan son atendidos, es decir, (First Input First Output, FIFO).
- El "cambio de contexto" sólo ocurre cuando el proceso termina y no hay reorganización de los procesos encolados.
- La carga del planificador es mínima.

No hay concurrencia.

Pobre respuesta en procesos interactivos.

Schedulers

1. Algoritmo FCFS (First Come, First Served)

Funcionamiento resumido

- El tiempo de una ronda (*turn around*), tiempo de espera y tiempo de respuesta puede ser alto por las mismas razones.
- No se tienen **prioridades**, por lo que el sistema tiene problemas para tener procesos que cooperan (meeting) al mismo tiempo (deadlines).
- La falta de prioridades significa que mientras los procesos terminen eventualmente, no hay *starvation*.
- En un ambiente donde los procesos puede que no terminen (servidores), puede haber *starvation* (inalición).
- El ***throughput*** puede ser bajo debido a que procesos largos ocupan mucho tiempo el CPU.

Conceptos de los planificadores

Rendimiento (*Throughput*)

Definición

- El volumen de trabajo o información que fluye/tiene un sistema.
- En redes de datos se distingue entre:
  - Ancho de Banda (teórico del medio de transmisión),
  - Capacidad del canal (real máximo quitando encabezados, colisiones - probabilidad, Max. througput),
  - Flujo real en un momento dado o promedio (vel en Mbits por segundo).
- En los medios de almacenamiento (por ej. discos duros) pasa lo mismo, sobre todo si el sistema maneja redundancia.
- Como medimos la eficiencia? Caso x caso

Schedulers

2. Algoritmo SJF (Shortest Job First)

Funcionamiento resumido

- Está diseñado para maximizar el **throughput** en la mayoría de los escenarios.
- El planificador ordena los procesos utilizando la estimación del menor tiempo de procesamiento que le queda para terminar.
- Esto requiere conocimientos previos o estimaciones del tiempo requerido para que un proceso termine.
- Existen 2 variantes:
  - Preemptivo: Si un proceso corto llega durante la ejecución de otro, el proceso actual puede ser interrumpido, dividiendo el proceso en 2 bloques de cómputo separados.
  - Cooperativo: Se espera a que termine el proceso (como FCFS).

Schedulers

2. Algoritmo SJF (Shortest Job First)

Funcionamiento resumido

- Esto crea exceso de carga por los cambios de contexto adicionales.
- El planificador debe colocar cada proceso que llega en un lugar específico de la cola (creando carga adicional, inserción).
- El tiempo de espera y de respuesta se incrementan conforme incrementan los requerimientos de procesamiento.
- Dado que el tiempo de una ronda se basa en el tiempo de espera más el tiempo de procesamiento, procesos largos se afectan significativamente.

Schedulers

2. Algoritmo SJF (Shortest Job First)

Funcionamiento resumido

- El tiempo de espera total es menor que en el algoritmo FIFO
  - Esto apesar de que los procesos no esperen a que terminen los procesos largos.
- No se pone atención al *deadline*, así que el programador sólo puede intentar que los procesos con deadlines sean lo más cortos posibles.
- Starvation (inanición) es posible, especialmente en sistemas que ejecutan muchos procesos pequeños.

Schedulers

3. Algoritmo RR (Round Robin)

Funcionamiento resumido

- Objetivo principal (en general) es:
  - Dar buenos tiempos de respuesta (para entornos interactivos).
  - Compartir equitativamente los recursos del sistema entre todos los usuarios (para sistemas de tiempo compartido).
- La planificación más popular para este objetivo es Round Robin (división en el tiempo en fracciones):
  - El tiempo de CPU se divide en fracciones o cuantums que se asignan a los procesos.
  - Ningún proceso puede ejecutarse durante más de una fracción de tiempo habiendo procesos en espera.

Schedulers

3. Algoritmo RR (Round Robin)

Funcionamiento resumido

- Puede ocurrir que:
  - Un proceso necesite más de un cuantum
    El proceso pasará a la cola de procesos preparados.
  - Un proceso ceda el control de la CPU (ej, op. E/S)
    Se planifica otro proceso para que se ejecute.
- Cada proceso recibirá 1/N del tiempo de la CPU
  - N el número de procesos preparados.
- Los procesos cortos tendrán buenos tiempos de respuesta
  - Se pueden ejecutar en un único quantum de tiempo.

Schedulers

3. Algoritmo RR (Round Robin)

Funcionamiento resumido

¿Qué longitud debe tener el cuantum?

- Supongamos que la conmutación de un proceso a otro toma 5 ms: salvar y cargar registros, mapas de memoria,. etc.
- Supongamos que tmpo. del cuantum es de 20 ms.
- Tmpo. total de 25 ms.
- Esto significa que el 20% del tiempo de CPU lo gasta el S.O.
- Para mejorar la eficacia, se puede aumentar el tamaño del cuantum a 500 ms, con lo que el S.O. gasta el 1%.

Schedulers

3. Algoritmo RR (Round Robin)

Funcionamiento resumido

Problema (procesos interactivos)

- Si 10 usuarios pulsan <return> al mismo tiempo, entonces habrá 10 procesos en la lista de procesos ejecutables.
- Inmediatamente, uno de los procesos comenzará, el segundo tendrá que esperar aproximadamente ½ sg, ... y el último tendrá que esperar 5 seg. antes de tener una oportunidad, siempre y cuando todos los procesos agoten su cuantum.
- Es decir, el tiempo de respuesta de un comando corto es grande.

Schedulers

3. Algoritmo RR (Round Robin)

Conclusiones

- Si el cuantum es:
  - Pequeño: se conmutan muchos procesos y por tanto es menos eficiencia de CPU.
  - Grande: Respuesta pobre (delay) a demandas de usuarios interactivos.
- Solución de compromiso: tiempo de cuantum = unos 100 ms.
- Se supone que todos los procesos son de la misma importancia.

trade-off / Compromiso entre 2 cosas / Interacción o procesamiento

Schedulers

Investigar

Tarea obligatoria

1. Que valor o cómo se calcula el quantum en los actuales SOs?
   - Dar referencias válidas y recientes (no blogs)
   - Repartirse los SOs: Linux, Windows, MacOS, etc.

   Entregar un resumen de a lo más 2 cuartillas.

2. En Linux donde está el código del planificador?
   Identifcar los archivos, las funciones y las estructuras de datos.
   Entregar un resumen con el kernel más actual.

Conceptos de los planificadores

Planificador con prioridades

Funcionamiento resumido

1. Se usa cuando todos los procesos no tienen la misma importancia.
2. A cada proceso se le asigna una prioridad.
3. El proceso con la mayor prioridad es elegido y ejecutado (más tiempo).
4. El planificador debe ser justo: puede bajar o subir la prioridad. Para no obtener todo o casi nada del tiempo del CPU.
5. A menudo las prioridades están agrupadas en clases.
6. Cada proceso dentro de una clase de prioridad debe a su vez ser planificado (usando RR).

Schedulers

4. Algoritmo MLQ (Multi Level Queues)

Funcionamiento resumido

- Clasifica la carga de trabajo de acuerdo con sus características y mantiene colas separadas de procesos atendidos por distintos planificadores.
- Un ejemplo de división de la carga de trabajo podría ser:
    i. Procesos del sistema,
    ii. Programas interactivos, y
    iii. Trabajos por lotes.
- Un proceso se adscribe a una cola específica en virtud de sus atributos, que pueden ser suministrados por el usuario o por el sistema.

Schedulers

4. Algoritmo MLQ (Multi Level Queues)

Funcionamiento resumido

- La planificación en las colas puede ser:
  - Prioridad absoluta:
    Los procesos de la cola con la prioridad más alta se les da servicio hasta que la cola se quede vacía.
  - División del tiempo: con alguna variante que refleje la prioridad relativa de los procesos colocados en las colas específicas.
- Para procesos individuales:
  - Procesos interactivos se planifican por fracción de tiempo, y
  - Procesos por lotes (batch) se planifican por FCFS.

Schedulers

5. Algoritmo MLFQ (Multi Level Feedback Queues)

Funcionamiento resumido

- Es una variante de la planificación MLQ.
- Objetivo: dar tratamiento preferente a los procesos cortos y hacer que los que consumen recursos o largos se "hundan" lentamente en colas de nivel inferior para mantener elevada la utilización de la CPU.
- No se asignan clases fijas de procesos a colas específicas, es decir, el proceso pasa por las colas según el tiempo de ejecución:
  - Un proceso puede comenzar en la cola de nivel superior.
  - Si el proceso se completa en un cuantum sale del sistema.
  - Si necesita más tiempo para completarse, pasa a una cola con prioridad inferior.

Schedulers

5. Algoritmo MLFQ (Multi Level Feedback Queues)

Ejemplo

- Una variante de la organización sería:

Cola más alta	Se asigna 1 quantum
Cola siguiente	Se asignan 2 quantums
...	Se asignan 4 quantums
...	...

Schedulers

6. Algoritmo SRTF (Shortest Run-Time First)

Funcionamiento resumido

Es una versión más dinámica que SJF en la cual continua usando la estimación del tiempo de procesamiento requerido del proceso:

- Las estimaciones sirven para determinar sólo las prioridades iniciales.
- Cada vez que un proceso necesita parar o es interrumpido, el tiempo de procesamiento transcurrido hasta ese instante es restado del estimado.
- Cuando el proceso vuelve a estar preparado otra vez, adquiere una prioridad más alta, reflejando la realidad de que le queda menos trabajo.

Bibliography

Bibliografía

Algunas páginas a consultar sobre el tema son:

- http://en.wikipedia.org/wiki/Scheduling_(computing)
- https://www.cs.rutgers.edu/~pxk/416/notes/07-scheduling.html
- https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html

Algunos documentos

- https://web.cs.wpi.edu/~cs3013/c07/lectures/Section05-Scheduling.pdf

Case Study

Casos de Estudio

Process management on

Windows

DOS/Windows

Introducción

- Windows implements a **priority**-driven, **preemptive** scheduling system—the highest-priority runnable (*ready*) thread always runs, with the caveat that the thread chosen to run might be limited by the processors on which the thread is allowed to run, a phenomenon called *processor affinity*.
- The Windows scheduling code is implemented in the kernel.
  - There's no single "scheduler" module or routine, however—the code is spread throughout the kernel in which scheduling-related events occur.
  - The routines that perform these duties are collectively called the kernel's *dispatcher*.

DOS/Windows

Introducción

- By default, threads can run on any available processor, but you can alter processor affinity by using one of the Windows scheduling functions.
- When a thread is selected to run, it runs for an amount of time called a **quantum**:
  - A quantum is the length of time a thread is allowed to run before another thread at the same priority level (or higher, which can occur on a multiprocessor system) is given a turn to run.
  - Quantum values can vary from system to system and process to process for any of three reasons: system configuration settings (long or short quantums), foreground/background status of the process, or use of the job object to alter the quantum.

DOS/Windows

Administrador de Procesos

Bibliografía

- Book Windows Internals, 6th edition
  Mark Russinovich & David Solomon
  https://technet.microsoft.com/en-us/sysinternals/bb963901.aspx
- Web
  Windows Sysinternals
  https://technet.microsoft.com/en-us/sysinternals/default.aspx
- Scheduler
  - https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=7

DOS/Windows

Administrador de Procesos

Herramientas

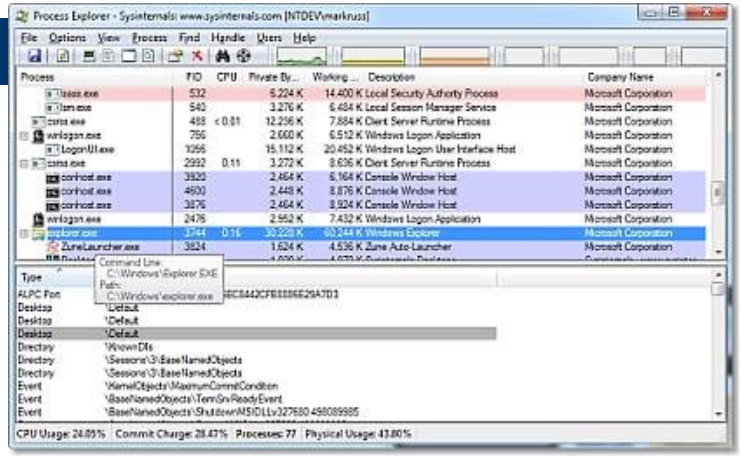
Existen varias aplicaciones de Microsoft y externas como las siguientes:

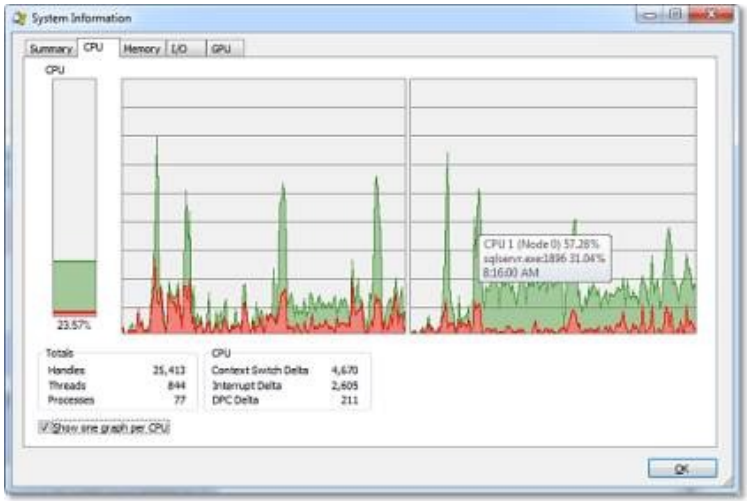
- Process Explorer

  Permite ver varios elementos como threads, variables del ambiente, strings, protocolos/puertos, etc. Además tiene la opción de ver información general del sistema

- Process Hacker

  Similar al anterior pero permite ver más datos.

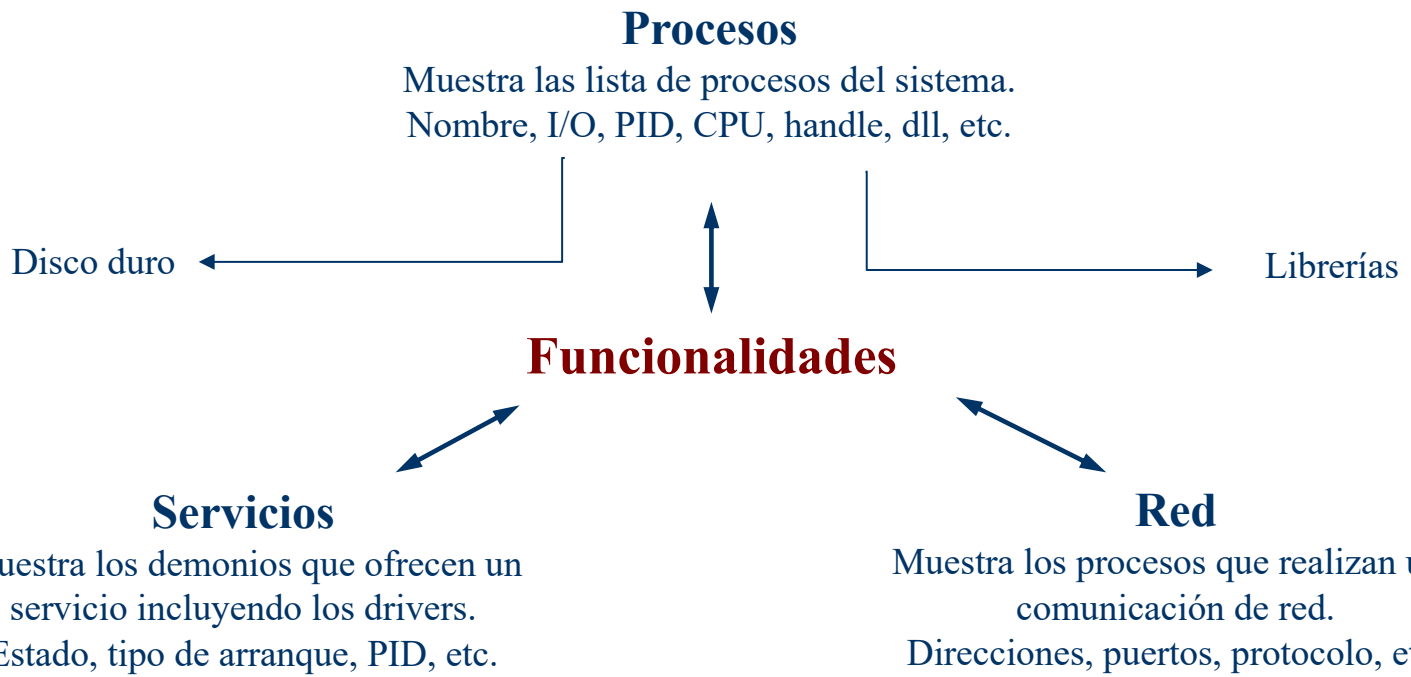




DOS/Windows

Administrador de Procesos

Mapa mental



Comparativo
https://chentiangemalc.wordpress.com/2011/06/13/process-explorer-vs-process-hackerpart-1-of-2/

Case Study

Casos de Estudio

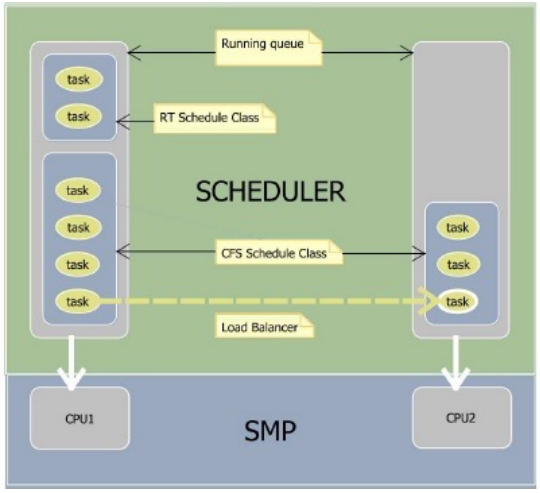
Process management on

UNIX-like

Linux

UNIX

Introducción



El administrador de procesos de UNIX se caracteriza por:

- Manejar prioridades [-20, 19] negativas más prioritarios.
- Un scheduler de tipo CFS (*Completely Fair Scheduler*).

Ha evolucionado:

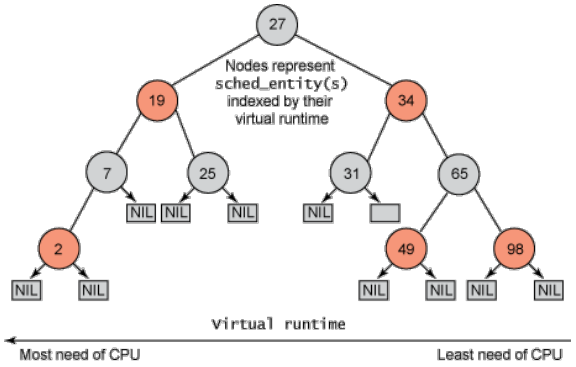
- Para adaptarse al uso de varios procesadores (SMP).
- De un esquema de colas (run queues) a uno de árbol (red-black tree).
- Cambiar la jerga: Timeslices a Timeline.
- Se adecuó a las necesidades de los procesos cambiando de complejidad: lineal vs logarítmica.
- Manejar procesos e hilos, incluso en el kernel.

Linux

Administrador de procesos

Linux Scheduler

- O(n).- Kernel 2.4 -2.6.22
- O(1).- Kernel 2.6.23. Desarrollado por **Ingo Molnar**.
- O(log N).- CFS (Completely Fair Scheduler).



Trabaja en Red Hat en:
Sistemas de Tiempo Real
Entre Linux 2.6.21 y Linux 2.6.24, trabajó en el Completely Fair Scheduler (CFS)
Se inspiro en el trabajo de Con Kolivas

Con Kolivas desarrollo el software ConTest para medir el rendimiento de kerneles de Linux. Última actualización 2003.

Linux

Administrador de procesos

Comandos

$ uname –p # Datos del procesador
$ cat /proc/cpuinfo
$

Linux

Rendimiento del kernel

Un sistema operativo (kernel) debe ser bastante eficiente así que existen bastantes trabajos que intentan medir su rendimiento (performance) sobre todo en micro-kernels.

Papers / Benchmarks

- https://os.inf.tu-dresden.de/pubs/sosp97/
- https://openbenchmarking.org/s/Kernel
- http://events.linuxfoundation.org/sites/events/files/slides/dbueso-lpc-2015-kperfmonitor.pdf

Linux

Temas avanzados

- Symmetric Multiprocessor (SMP)
  - https://en.wikipedia.org/wiki/Symmetric_multiprocessing
  - https://www.ibm.com/developerworks/library/l-linux-smp/
  - https://www.ibm.com/developerworks/linux/library/l-scheduler/

- Frameworks
  - Bossa - A Framework for Scheduler Development.

    Linux 2.6 trabajos en el 2005

    Linux 2.4 trabajos en el 2004

    http://bossa.lip6.fr/

- Inside the Linux 2.6 CFS
  - http://www.ibm.com/developerworks/library/l-completely-fair-scheduler/index.html

Tarea

Análisis del kernel de Linux

- Identificar el código del Administrador de Procesos
  - Describir su estructura: datos y funciones.
  - Las variables del kernel (/proc)
  - Donde está el planificador (scheduler).
  - Cuales son las rutinas de cambio de contexto (ensamblador).
- Identificar el código de Procesos e Hilos
  - Donde está el API
- Explorar el código (local y web)
  - http://git.kernel.org
  - http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/kernel/sched?id=HEAD.
- Como hacer un análisis automático del código?
  - Jerarquía de funciones
  - Parámetros, etc.
- Usar la documentación oficial y más reciente
  - https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt
  - http://man7.org/linux/man-pages/man7/sched.7.html
  - https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Performance_Tuning_Guide/s-cpu-scheduler.html

Conclusiones

Adm Procesos

Conceptos a recordar

- Lluvia de ideas.

The end

Contacto

Raúl Acosta Bermejo

http:www.cic.ipn.mx
http://www.ciseg.cic.ipn.mx/

racostab@ipn.mx
racosta@cic.ipn.mx

57-29-60-00
Ext. 56652