Diseño de Sistemas Operativos

David Pérez

david.perez@ciens.ucv.ve

Asignación - Hilos

- Concepto de Hilo
- ¿Posee alguna estructura?
  - Si o No
  - Si la respuesta es positiva
    - ¿Qué debe contener la estructura de control del hilo?

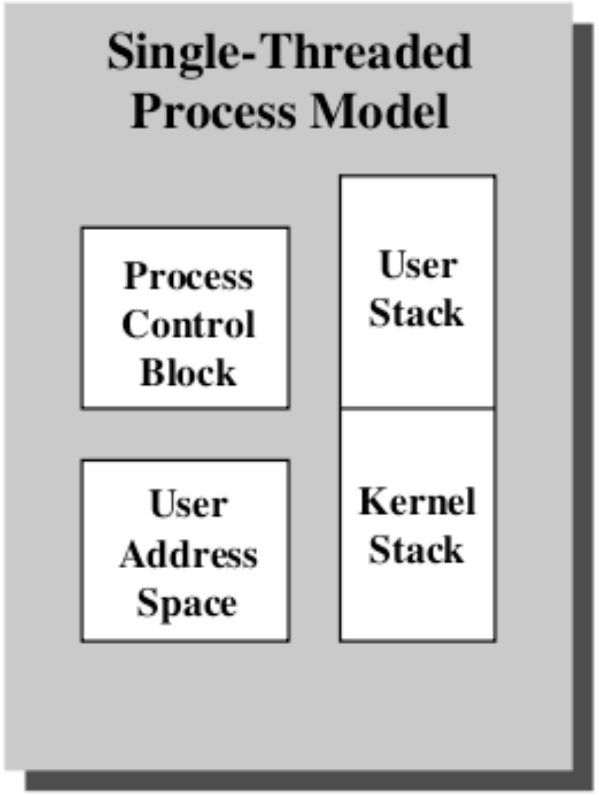
Asignación - Hilos

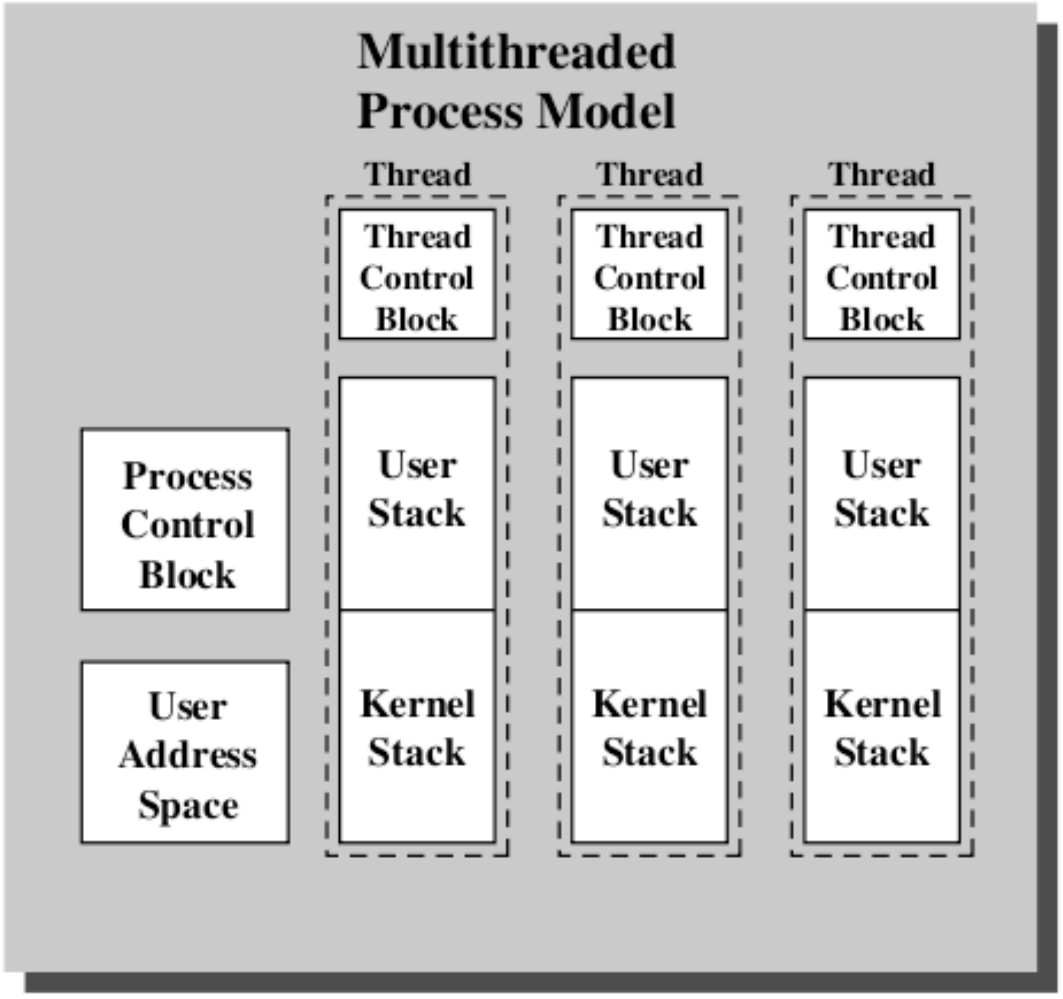
- Un proceso tiene asociado
  - Imagen del proceso
    - Espacio de direcciones virtuales
  - Acceso protegido
    - Procesador
    - Procesos
    - Archivos
    - E/S

Asignación - Hilos

- Un hilo debe poseer
  - El estado de ejecución del hilo
  - El contexto del procesador
  - Una pila de ejecución
  - Almacenamiento estático
  - Acceso a memoria y a los recursos del proceso

Asignación - Hilos





Asignación - Hilos

- ¿Qué comparten los hilos de un proceso?
  - Estado del proceso
  - Recursos del proceso
  - Espacio de direcciones
  - Acceso a los mismos datos

Diseño de Sistemas Operativos

- Es diferente estudiar sobre un sistema operativo que diseñarlo e implementarlo
- Creencias más no estándares

Diseño de Sistemas Operativos

- *The Mythical Man Month – Fred Brooks (1975)*
- *Operating System: A Design-Oriented Approach – Crowley (1997)*
- *Hints for Computer System Design – Lampson (1984)*
- *On Building Systems That Will Fail – Corbató (1991)*

Diseño de Sistemas Operativos

- La naturaleza del problema del diseño
  - Metas
  - Diseño de interfaces
  - Implementación
  - Desempeño
  - Administración de proyectos
  - Tendencias del diseño

Diseño de Sistemas Operativos

- Metas
  - Ideas claras
  - ¬ ideas --> Dificultades
    - ¿Dónde?
  - Ejemplo
    - PL/I y C (1960)
    - PL/I --> Fortran, COBOL, Algol
    - C --> Dennis Ritchie

Diseño de Sistemas Operativos

- Metas
  - Tener una visión clara es crucial
  - ¿Qué quieren los diseñadores de SO?
    - Ideas

Diseño de Sistemas Operativos

- Metas
  - SO de propósito general
    - Definir abstracciones
    - Proporcionar operaciones primitivas
    - Garantizar el aislamiento
    - Administrar el hardware

Diseño de Sistemas Operativos

- Metas
  - SO de propósito general
    - Definir abstracciones
    - Proporcionar operaciones primitivas
    - Garantizar el aislamiento
    - Administrar el hardware

Diseño de Sistemas Operativos

- Definir abstracciones
  - ¿Qué es una abstracción?
  - Ejemplos de abstracción en un SO
  - Sincronización
  - Modelo de memoria
- Abstracciones vía estructuras de datos

Diseño de Sistemas Operativos

- Proporcionar operaciones primitivas
  - Manipular estructuras de datos
    - ¿Proporcionadas por quien?
  - Llamadas al sistema
  - Para el usuario
    - SO
      - Abstracción
      - Llamadas al sistema

Diseño de Sistemas Operativos

- Garantizar el aislamiento
  - Múltiples usuarios
  - ¿Qué hacer?
  - Ejecuciones autorizadas
    - Instrucciones autorizadas
    - Datos autorizados
  - ¿Cómo hacer la compartición?
  - Aislamiento selectivo
    - Ejemplo

Diseño de Sistemas Operativos

- Administrar el hardware
  - Es obvio
  - ¿Por qué?

¿Por qué es difícil diseñar SO?

- Ley de Moore
  - El hardware de computadora mejora en un factor de 100 cada década.
- Sin embargo
  - Nadie afirma que los sistemas operativos mejoran en un factor de 100 cada década
  - Ni siquiera que mejoran
  - Tal vez las versiones antiguas sean más confiables que las actuales

¿Por qué es difícil diseñar SO?

- ¿Por qué ocurre lo anterior?
  - Inercia
  - Compatibilidad con sistemas anteriores
- ¿Será un sistema operativo, igual a un programa de aplicación?
- ¿Ambos podrían verse como procesos?

¿Por qué es difícil diseñar SO?

- ¿Qué diferencia a un SO de un programa de aplicación?
- Top eight

¿Por qué es difícil diseñar SO?

- 1°
  - Los SO son extremadamente grandes
    - ¿Cuanto tiempo les tomaría hacer un SO serio?
    - UNIX
      - Por encima de un millón de líneas
    - Windows
      - 29 millones de líneas de código
  - Tampoco exageremos
    - ¿Un portaaviones será más complicado que un SO?

¿Por qué es difícil diseñar SO?

- 2°
  - Manejo de concurrencia
    - Múltiples usuarios y múltiples dispositivos de E/S
    - Condiciones de competencia
    - Deadlocks

¿Por qué es difícil diseñar SO?

- 2°
  - Manejo de concurrencia
    - Múltiples usuarios y múltiples dispositivos de E/S
    - Condiciones de competencia
    - Deadlocks

¿Por qué es difícil diseñar SO?

- 3°
  - Usuarios hostiles
  - Complicaciones de acceso
  - ¿Un editor de texto sufrirá estas consecuencias?

¿Por qué es difícil diseñar SO?

- 4°
  - Compartición de recursos
    - ¿Qué pasa con los usuarios malintencionados?
    - ¿Estos es una contradicción?

¿Por qué es difícil diseñar SO?

- 5°
  - Vida larga
    - Unix --> + 25 años
    - Windows --> + 10 años
  - Consideraciones
    - Cambios en el hardware
    - Cambios en las aplicaciones
  - Los sistemas casados con una visión específica del mundo tienden a desaparecer

¿Por qué es difícil diseñar SO?

- 6°
  - No se tiene un idea clara
    - ¿De qué?
  - Alto grado de generalidad
  - Ejemplos
    - SO
    - Barcos

¿Por qué es difícil diseñar SO?

- 7°
  - Portabilidad
    - Múltiples arquitecturas
      - Big-Endian
      - Little-Endian
      - Diferencias
    - Múltiples dispositivos
      - IRQs (Interrupt ReQuest) repetidos

¿Por qué es difícil diseñar SO?

- 8°
  - Compatibilidad
    - ¿Hacia dónde?
    - ¿Por qué?
  - Ejemplo
    - Computacional
    - Carros

Diseño de Interfaces

- Si diseñar es complicado, ¿por donde empezar?
- Interfaces
  - Conjuntos de servicios
    - Tipos de datos
      - Archivo
    - Operaciones sobre los tipos de datos
      - Read
- ¿Quien es el usuario para el SO?

Diseño de Interfaces

- Interfaces de llamadas el sistema
- Interfaces de bajo nivel
  - Ejemplos

Diseño de Interfaces

- Principios Orientadores
  - Sencillez
    - Fácil entendimiento
    - Fácil implementación
  - *“Se alcanza la perfección no cuando ya no queda más que añadir, sino cuando ya no queda más que quitar” -- Antonie de Saint-Exupupéry*
  - *Menos es mejor que más*
  - *Principio KISS (Keep It Simple, Stupid)*

Diseño de Interfaces

- Principios Orientadores
  - Integridad
    - Hacer lo que se debe hacer
      - Completa o Integra
  - *"Todo de ser lo más sencillo posible, pero no más"*
    -- Albert Einsten
  - Funcionalidades bien definidas y correctas

Diseño de Interfaces

- Principios Orientadores
  - Integridad
  - *“En primer lugar, es importante destacar el valor de la sencillez y la elegancia, pues la complejidad suele multiplicar las dificultades y, como hemos visto, propiciar errores. Mi definición de elegancia es el logro de una funcionalidad dada con un mínimo mecanismo y un máximo de claridad” -- Fernando Corbato*
  - Idea clave
    - Mínimo mecanismo

Diseño de Interfaces

- Principios Orientadores
  - Integridad
  - Cada característica, función y llamada al sistema debe sostener su propio peso
  - Solo una cosa, pero se hace bien
  - Debo preguntarme algo antes de incluir una funcionalidad a SO
  - ¿Sucedería algo terrible si no incluyéramos la funcionalidad?
    - Si o No

Diseño de Interfaces

- Principios Orientadores
  - Integridad
  - Ejemplos
    - MINIX
      - sendsend, receive y sendrec
      - ¿Se les parece a algún esquema estudiado?
    - Amoeba
      - Llamada a procedimiento remoto

Diseño de Interfaces

- Principios Orientadores
  - Eficiencia
    - Rápido y bien
    - Nada de marramucias
    - Es ocasiones es mejor no implementar, si no existe eficiencia
    - Se requiere preservar intuición por parte del programador
      - Ejemplos
        - » lseek vs. read

Diseño de Interfaces

- Paradigmas
  - Ya con metas, comenzamos con el diseño
  - Buen punto de partida
    - ¿Cómo los clientes verán al sistema?
  - Ofrecer un conjunto consistente de características
    - Coherencia arquitectónica

Diseño de Interfaces

- Tipos de usuarios
  - Interacción con los programas de ejecución
    - ¿Con qué interactúan?
  - Construyen o programan dichos programas
    - ¿Con qué interactúan?
  - Ejemplos
    - Macintosh
    - UNIX

Diseño de Interfaces

- Si creamos, ¿cómo vamos?
  - Crear la GUI
    - Modelo descendente
    - ¿Qué debo saber?
      - Orientado a eventos
      - Orientado a manejo de caracteres
  - Crear la interfaz de llamadas al sistema
    - Modelo ascendente
      - Características ofrecidas a los programadores
      - Ejemplo --> X en Linux

Diseño de Interfaces

- Paradigmas de interfaz de usuario
  - Buen paradigma
    - Metáforas
    - ¿Qué significa esto?
    - ¿Qué facilita?
    - Implicaciones
      - Bibliotecas y herramientas
      - Manejo uniforme

Diseño de Interfaces

- Paradigmas de ejecución
  - ¿Qué se pretende mantener?
  - Paradigma algorítmico
    - Ideas
  - Paradigma controlado por sucesos
    - Ideas
  - ¿Qué provee el sistema operativo en cada caso?

Diseño de Interfaces

```
main()
{
        int ...;
        init();
        hacer_algo();
        read(...);
        hacer_otra_cosa();
        write(...);
        seguir_trabajando();
        exit(0);
}
```

```
main()
{
        mees_t mens;
        init();
        while(rec_mensaje(&mens)){
                switch(msg.type){
                        case 1: ...;
                        case 2: ...;
                        ....
                }
        }
}
```

Diseño de Interfaces

- Paradigmas de datos
  - ¿Cómo se presentan los dispositivos y las estructuras al programador?
    - FORTRAN
      - Todos es una cinta
    - UNIX
      - Todos es un archivo
    - Windows
      - Todo es un objeto
    - WEB
      - Todo es un documento

Diseño de Interfaces

- Paradigmas de datos
  - Ejemplos
    - fd1 = open (“file1”,O_RDWR);
    - fd2 = open (“/dev/tty”,O_RDWR);
    - fd3 = open (“/proc/501”,O_RDWR);

Diseño de Interfaces

- La interfaz de llamada al sistema
  - Conjunto de llamadas al sistema
    - Lo más sencillo posible, pero no más sencilla
  - ¿Servirá de algo el paradigma de datos acá?
    - Ideas

Diseño de Interfaces

- La interfaz de llamada al sistema
  - Ejemplos
    - exec (nombre,argp,envp)
    - execl (nombre,arg0,arg1,...,argn,0)
    - execle (nombre,arg0,arg1,...,argn,envp)
    - Sin exagerar
      - ¿Por qué?

Diseño de Interfaces

- La interfaz de llamada al sistema
  - Creación de procesos
    - Linux
    - Windows
    - Primera ley de Tanenbaum
      - “Añadir más código añade más errores”

Diseño de Interfaces

- La interfaz de llamada al sistema
  - Consideración adicional – Lampson (1984)
    - “No hay que ocultar la potencia”
    - ¿Qué significa los anterior?
    - Ejemplos
      - Linux vs. Windows
      - Lecturas de la RAM de vídeo

Implementación

- Estructura del sistema
  - Decidir la estructura que adoptará el sistema
  - Opciones
    - Capas
    - Exokernel
    - Cliente – Servidor (Microkernel)
    - Sistemas Extensibles
    - Subprocesos del kernel

Implementación

- Mecanismo vs. Políticas
  - Mecanismo -- ¿En donde?
  - Políticas -- ¿En donde?
  - Ejemplos
    - Planificación
    - Memoria
    - Carga de módulos

Implementación

- Ortogonalidad
  - Definición
  - Ejemplo
    - Lenguaje C
  - Es consecuencia directa de los principios
    - Sencillez
    - Integridad

Implementación

- Ortogonalidad
  - Ejemplos
    - Clone – Linux
    - Procesos y Subproceso – Windows 2000

Implementación

- Asignación de nombres
  - ¿Cómo referenciar: objetos, recursos, archivos?
  - Creación y administración de nombre
    - Clave
  - Jerarquías
    - Ejemplos

Implementación

- Asignación de nombres
  - Niveles
    - Externo
    - Interno
  - Ejemplo
    - Windows
      - Nombres de archivos, nombres de objetos, nombres del registro, nombre active directory.

Implementación

- Asignación de nombres
  - ¿Que considerar?
    - Cantidad de espacios de nombre
    - Sintaxis
    - Relativos
    - Absolutos

Implementación

- Tiempo de enlace
  - ¿La correspondencia entre nombres y objetos es fija?
    - Si o No
  - ¿Qué es enlazar?
    - Enlace temprano
    - Enlace tardío

Implementación

- Tiempo de enlace
  - Ejemplos
    - Universidad
    - Fabrica
    - Lenguajes de programación
      - Variable global
      - Variable local
      - Malloc o New

Implementación

- Tiempo de enlace
  - ¿Qué tipo de enlace utilizar en un SO?
    - Ideas
  - Ejemplos
    - Memoria
    - GUI

Implementación

- Estructuras estáticas o dinámicas
  - Siempre nos vemos obligados a escoger
  - ¿Por qué?
  - Ventajas
  - Desventajas

Implementación

- Estructuras estáticas o dinámicas
  - Ejemplos
    - Tablas de procesos
    - Tablas de archivos abiertos
  - Soluciones
    - Ideas

Implementación

- Estructuras estáticas o dinámicas
  - Soluciones
    - Listas enlazadas
      - Funcionamiento
    - Ejemplo --> Tabla de procesos
    - ¿De qué tamaño podría ser la tabla de procesos?
    - De hecho
    - ¿De qué tamaño podría ser cualquier tabla?

Implementación

- Estructuras estáticas o dinámicas
  - Búsquedas en las estructuras
    - Ejemplo
      - Buscar un proceso por PID

Implementación

- Estructuras estáticas o dinámicas
  - Búsquedas en las estructuras

```
found = 0;
for (p=&proc_table[0]; p<&proc_table[PROC_TABLE_SIZE]; p++){
        if(p->proc_pid == pid){
                found=1;
                break;
        }
}
```

Implementación

- Estructuras estáticas o dinámicas
  - Tablas estáticas
    - ¿Cuando es recomendable usarlas?
      - Ejemplos
        - SO monousuarios
        - Conocimiento con antelación
  - Otra alternativa
    - Fijo, pero no…
    - Ideas

Implementación

- Estructuras estáticas o dinámicas
  - Algunas consideraciones sobre el kernel parecidas a este enfoque
    - Pilas
    - Planificación de procesos
    - Estructura del kernel

Implementación

- Descendente o Ascendente
  - Top-Down
    - Manejadores de llamadas al sistema
      - Mecanismos y Estructuras
    - Continuo descendiendo
    - Consideraciones
      - Ideas
      - ¿Algún inconveniente?

Implementación

- Descendente o Ascendente
  - Bottom-up
    - Ocultar el hardware --> HAL
    - Manejador de interrupciones
    - Controlador de reloj
    - Planificador
    - Tablas y estructuras de datos
    - Protección
    - ¿Y la E/S?

Implementación

- Descendente o Ascendente
  - ¿Existe otro modelo?
    - Ideas
    - Ejemplos
    - Consideraciones

Implementación

- Técnicas útiles
  - Ocultación del hardware
    - Una buena parte del hardware es fea
    - Ocultarla desde el principio
    - ¿En qué caso no?
    - HAL
    - ¿Cómo manejar las interrupciones?
      - Ideas

Implementación

- Técnicas útiles
  - ¿Cómo manejar las interrupciones?
    - Convertirlas de inmediato
    - Convertirlas en una operación unlock
    - Convertirla en un mensaje
  - ¿Características en común?
    - Ideas

Implementación

- Técnicas útiles
  - Ocultación del hardware
    - Múltiples plataformas
      - Tamaño de RAM
      - CPU
      - Tamaño de la palabra
    - ¿Cómo ocultar dichas diferencias?
      - Ideas para item

Implementación

- Técnicas útiles
  - Ocultación del hardware
    - Compilación condicional

```
#include “config.h”
init()
{
    #if(CPU==PENTIUM)
    #endif

    #if (CPU==ULTRASPARC)
    #endif
}
```

Implementación

- Técnicas útiles
  - Indirección
    - ¿Qué es esto?
    - Se dice que no hay un problema en la computación que no pueda resolverse con un nivel adicional de indirección
    - Ejemplo
      - Teclas --> Sistemas basados en Pentium
      - Caracteres ASCII --> Fuentes
      - Buzones
      - Macros
        - #define PROC_TABLE_SIZE 256

Implementación

- Técnicas útiles
  - Reusabilidad
    - Concepto
    - Beneficios
    - Ejemplos
    - ¿Mismo contexto?

Implementación

- Técnicas útiles
  - Reentrancia
    - Diferentes filosofías
      - Modificar vs. Ejecutar -- ¿Alguna relación?
    - Ejecución de un fragmento de código --> Instancias simultaneas
    - Ejemplos
      - Sistemas multiprocesador
      - Sistemas uniprocesador
        - Consideraciones
    - Solución
      - SO reentrante
      - Estructuras protegidas por mutex
      - Inhabilitar interrupciones

Implementación

- Técnicas útiles
  - Reentrancia
    - Ejemplo
      - Planificador
        1. Un proceso agoto su tiempo
        2. El SO lo pasa al final de la cola
        3. Ocurre una interrupción -- Proceso pasa a listo
        4. ¿Y entonces?

Implementación

- Técnicas útiles
  - Fuerza bruta
    - A veces optimizar no es lo más recomendable
    - Ejemplos
      - Búsqueda en tablas
      - Conmutación de contexto

Implementación

- Técnicas útiles
  - Fuerza bruta
    - A veces optimizar no es lo más recomendable
    - Ejemplos
      - Búsqueda en tablas
      - Conmutación de contexto

Implementación

- Técnicas útiles
  - Verificar primero si hay errores
    - Muchas llamadas al sistema podrían fallar
      - Ejemplos
    - SO debe verificar minuciosamente todos los posibles errores antes de invocar o ejecutar cualquier llamada
    - Básicamente, realizar las pruebas al principio del procedimiento que ejecuta la llamada al sistema
    - ¿Por qué?
      - Asignación de recursos
    - ¿Qué pasa si no verifico?

Implementación

- Técnicas útiles
  - Verificar primero si hay errores
    - Ejemplos
      - malloc
      - free
    - ¿Quién podría ayudar acá?

Implementación

- Técnicas útiles
  - Verificar primero si hay errores
    - Ejemplos
      - malloc
      - free
    - ¿Quién podría ayudar acá?

Desempeño

- En condiciones normales
  - ¿Qué es mejor?
    - SO rápido vs. SO lento
  - Consideraciones
  - Implicaciones del desempeño

Desempeño

- ¿Por qué son lentos los sistemas operativos?
  - Generalmente la lentitud es culpa del propio SO
  - Ejemplos
    - Tiempo de carga: MS-DOS y UNIX Versión 7
    - Tiempo de carga: Windows Vista y Fedora Core 7
    - Consideraciones
  - Más de la cuenta
    - Ejemplo
      - Plug and Play
      - ¿Alguna solución adicional?

Desempeño

- ¿Por qué son lentos los sistemas operativos?
  - ¿SO amigables o a prueba de idiotas?
  - Opción --> Ser aún más selectivo
    - ¿Esto gustaría a los usuarios?
    - ¿Esta característica justifica el precio en cuanto a tamaño del código, lentitud, complejidad y confiabilidad?
  - También influye el marketing

Desempeño

- ¿Qué debe optimizarse?
  - La primera versión debe ser lo más directa posible
  - Solo optimizar lo que se está seguro dará problemas
  - Período de prueba
    - El estudio del período da como resultado nuevas optimizaciones
  - Anécdota
    - mkfs en MINIX
    - ¿Valió la pena?
  - *“Lo bastante bueno es bastante bueno”*
  - Otras consideraciones

Desempeño

- ¿Qué debe optimizarse?
  - Equilibrio espacio-tiempo
    - Sacrificar tiempo a uno para ganar el otro
    - Sustituir procedimientos pequeños por macros
    - Ventajas

Desempeño

- ¿Qué debe optimizarse?

```
#define BYTE_SIZE 8
int bit_count(int byte)
{
    int i, count=0;
    for(i=0;i<BYTE_SIZE;i++)
            if((byte>>1)&1) count++;
    return(count);
}
```

Desempeño

- ¿Qué debe optimizarse?

```
#define bit_count(b)
    (b&1)+((b>>1)&1)+((b>>2)&1)+((b>>3)&1)+((b>>4)&1)+…+((b>>7)&1
```

  - ¿Se diferencian en algo?
    - Ideas

Desempeño

- ¿Qué debe optimizarse?
  - Uso de cachés
    - Mejorar desempeño
    - ¿Dónde es aplicable?
    - Ejemplo
      - /usr/ast/correo
      - ¿Cuál es el proceso?
    - ¿Cómo mejorar lo anterior?

Desempeño

- ¿Qué debe optimizarse?
  - Uso de cachés

Ruta	Número de Nodo-i
/usr	6
/usr/ast	26
/usr/ast/correo	60

Desempeño

- ¿Qué debe optimizarse?
  - Sugerencias en vez de caches
    - ¿Qué es esto?
    - Ejemplo
      - URLs

Desempeño

- ¿Qué debe optimizarse?
  - Aprovechamiento de la localidad
    - Los proceso no actúan al azar
    - Principio de localidad de referencia
    - Ejemplos
      - Archivos --> Fast System de Berkeley (McKusick et al, 1984)
      - Planificación en sistemas multiprocesadores

Desempeño

- ¿Qué debe optimizarse?
  - Aprovechamiento de la localidad
    - Los proceso no actúan al azar
    - Principio de localidad de referencia
    - Ejemplos
      - Archivos --> Fast System de Berkeley (McKusick et al, 1984)
      - Planificación en sistemas multiprocesadores

Desempeño

- ¿Qué debe optimizarse?
  - Optimización del caso común
    - Se estudia el mejor, el peor y promedio de los casos
    - Se desea optimizar el caso más factible

Administración de Proyectos

- Los programadores son optimistas perpetuos
- Correr al teclado y empezar a escribir
- Luego depurar y correr
- En un SO esto no funciona

Administración de Proyectos

- El mes hombre mítico
  - Fred Brooks -- Libro -- OS/360
  - Programador --> 1000 líneas de depuradas al año
  - ¿Estará este hombre loco?
  - ¿Cómo podría ser esa la producción anual de una persona con un coeficiente intelectual mayor a 50?
  - Los proyectos grandes no son ni parecidos a proyectos completos

Administración de Proyectos

- El mes hombre mítico
  - Proyecto grande
    - Planificación
    - Especificaciones de módulos e interfaces
    - Interacción entre dichas partes
    - Codificación
    - Etc.

Administración de Proyectos

- El mes hombre mítico
  - Brooks estimo lo siguiente:
    - 1/3 Planificación
    - 1/6 Codificación
    - 1/4 Prueba de módulos
    - 1/4 Prueba del sistema
  - En otras palabras el código es lo fácil

Administración de Proyectos

- El mes hombre mítico
  - ¿Qué significa el título?
    - Las personas y el tiempo no son intercambiables
    - No existe unidad mes-hombre
    - 15 personas --> 2 años
    - 360 personas --> 1 mes
    - 60 personas --> 6 meses
    - *“La adición de personal a un proyecto de software atrasado lo atras más”*
    - *“Se necesitan 9 meses para tener un hijo, por más mujeres que se asignen a la tarea”*

Administración de Proyectos

- Estructura de equipos de trabajo
  - La calidad es inmensamente importante
  - 10 a 1 el factor de productividad de un programador experto a un rooki
  - El problema es el número de programadores expertos
  - Necesidad de mantener coherencia arquitectónica
  - Harlam Mills
    - Equipo de programador en jefe
    - *“Equipo de cirujanos, no de carniceros”*
    - Ejemplo de dicho enfoque

Administración de Proyectos

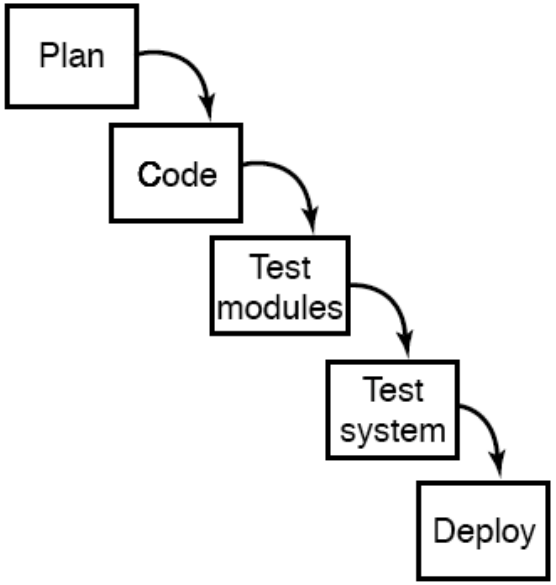
- Estructura de equipos de trabajo

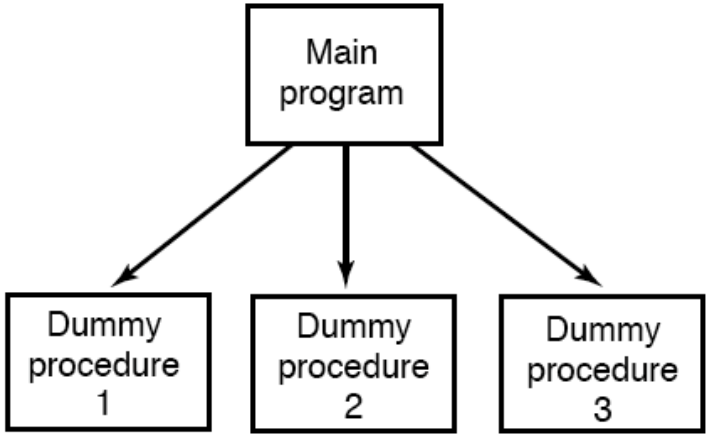
Title	Duties
Chief programmer	Performs the architectural design and writes the code
Copilot	Helps the chief programmer and serves as a sounding board
Administrator	Manages the people, budget, space, equipment, reporting, etc.
Editor	Edits the documentation, which must be written by the chief programmer
Secretaries	The administrator and editor each need a secretary
Program clerk	Maintains the code and documentation archives
Toolsmith	Provides any tools the chief programmer needs
Tester	Tests the chief programmer's code
Language lawyer	Part timer who can advise the chief programmer on the language

Fig. 12-9. Mills' proposal for populating a 10-person chief programmer team.

Administración de Proyectos

- El papel de la experiencia
  - Contar con diseñadores expertos es crucial
  - Romper con los paradigmas clásicos de desarrollo
  - Efecto segundo sistema





Administración de Proyectos

- No hay una bala de plata
  - Brooks
    - *“No Silver Bullet”*
    - Ningún remedio propuesto, provee una solución completa
    - ¿Cierto o falso?

Resumen

- Primero debo saber lo que voy a hacer
- Interfaces sencillas, completas y eficientes
- Paradigmas claros
  - Usuario
  - Ejecución
  - Datos
- Estructuración conocida
  - Capas, microkernel, etc.

Resumen

- Componentes internos ortogonales
- Separar mecanismos de políticas
- Consideraciones
  - Estructuras dinámicas
  - Estructuras estáticas
  - Asignación de nombres
  - Tiempo de enlace

Resumen

- Desempeño
- Administración del equipo
- ¿Será complejo diseñar, implantar y probar un SO?