1-Programas y servicios

Un programa es un proceso en ejecución, se encuentran en la memoria secundaria en un fichero con un formato definido en función del sistema operativo, el fichero en su i-nodo tiene un atributo que lo identifica como ejecutable, pudiendo ser ejecutado por un grupo, propietarios, resto usuario, en función de los permisos.

Información en un i-nodo:

Permisos(determina si es ejecutable), número de enlaces(determina cuantos nombres apuntan a este i-nodo, UID,GID, Tamaño, fecha, etc.

Como crear un ejecutable:

1. Creamos el código fuente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este archivo contiene el código fuente, se guarda con extensión .c.

Al guardar el archivo , el sistema crea un nuevo i-nodo para hola.c el cual guarda la información mostrada arriba.

1. Invocación del compilador gcc.

Abrimos una terminal y ejecutamos el siguiente comando

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Gcc es el compilador

-c le indica a gcc que solo compile y no lo enlace

Programa c es el archivo de código fuente

1. Comienza el preprocesamiento del código por ejemplo:

Se reemplaza la línea #include<stdio.h> por el contenido de stdio.h para poder leer printf entre otras cosas.

1. Después, gcc toma el código resultante y lo traduce a un archivo objeto, llamado programa.o, este archivo contiene el código en instrucciones de máquina que el procesador puede entender pero que no es ejecutable por sí solo, tiene un formato binario no legible por humanos contiene el código máquina.
2. Para completar el proceso invocamos al enlazador:

Texto

Descripción generada automáticamente

Programa.o es el archivo objeto

-o programa indica el nombre del archivo ejecutable que queremos crear en este caso programa.

El enlazador toma el archivo objeto programa.o y lo une con las librerías necesarias, el resultado final es un ejecutable llamado programa,este proceso resuelve todas las referencias a funciones y variables externas que el programa necesita para funcionar.

1. Lo ejecutamos con

Texto

Descripción generada automáticamente

Cuando el pedimos al SO que ejecute el programa, el SO activa el loader para :

1. Cargar el archivo ejecutable en memoria y copia sus segmentos(texto,datos y otros).
2. Se reserva espacio en la memoria para la pila y variables globales.
3. El So crea una estructura de datos que representará al proceso en ejecución, asignándole un PID.
4. Finalmente el loader transfiere el control al segmento de código del programa donde se encuentra la función main. En este momento el programa comienza a ejecutarse como un proceso independiente.

En un sistema operativo, el **loader** es el módulo encargado de cargar el archivo ejecutable en memoria, iniciando el proceso. Este proceso se divide en tres segmentos principales:

1. **Segmento de texto**: Contiene las instrucciones ejecutables del programa.
2. **Segmento de datos**: Almacena variables globales y estáticas inicializadas al comienzo del proceso.
3. **Segmento de pila**: Creado dinámicamente por el sistema operativo, contiene los marcos de pila para cada función invocada (parámetros, variables locales y referencias para restaurar el estado anterior).

El sistema operativo maneja dos pilas: una para el modo usuario y otra para el modo kernel, donde se almacenan las llamadas al sistema en modo supervisor.

**UNIX** es un sistema de multiproceso que permite ejecutar varios procesos simultáneamente. El **planificador** del núcleo decide qué proceso usa la CPU en cada momento. Los procesos se crean mediante la llamada fork(), que genera un proceso hijo con un único proceso padre. Cada proceso tiene un identificador único (PID) que persiste durante toda su vida.

El **proceso PID 0** es especial, creado al iniciar el sistema, y, tras una llamada a fork(), se convierte en el proceso intercambiador, que gestiona la memoria virtual. El proceso hijo init (PID 1) es el primer proceso de usuario y se encarga de iniciar los demás procesos del sistema, siguiendo las configuraciones en /etc/inittab.

En sistemas operativos, se puede obtener información sobre los procesos en ejecución. En UNIX, el comando ps muestra esta información en columnas clave:

* **USER**: Usuario propietario del proceso.
* **PID**: Identificador único del proceso.
* **PPID**: Identificador del proceso padre.
* **C**: Uso de CPU en porcentaje.
* **SZ**: Tamaño virtual del proceso en memoria.
* **RSS**: Tamaño de la parte residente en memoria en KB.
* **PRS**: Procesador donde se ejecuta el proceso.
* **TTY**: Terminal asociado al proceso (si no, muestra ?).
* **STIME**: Hora de inicio del proceso.
* **STAT**: Estado del proceso, que puede ser:
  + **D**: Suspendido sin interrupción.
  + **R**: En ejecución o en cola.
  + **S**: Suspendido interrumpible.
  + **T**: Detenido.
  + **W**: Paginado (obsoleto desde el kernel 2.6).
  + **X**: Muerto.
  + **Z**: Zombie (difunto, no reclamado).
* **CMD**: Comando ejecutable del proceso.

**Ciclo de Vida de un Proceso: Estados y Transiciones Básicas**

Durante su ciclo de vida, un proceso pasa por diferentes **estados**, que representan su situación actual en el sistema:

1. **Ejecución en Modo Usuario**: El proceso se ejecuta en modo usuario y usa recursos limitados.
2. **Ejecución en Modo Supervisor**: Aquí, el proceso accede a recursos del sistema operativo (modo kernel).
3. **Listo para Ejecutar**: El proceso espera a que el planificador le asigne la CPU para ejecutar.
4. **Durmiendo en Memoria Principal**: El proceso está temporalmente inactivo pero cargado en la memoria principal.
5. **Esperando ser Cargado en Memoria**: El proceso está listo para ejecutarse, pero espera ser cargado en la memoria.
6. **Durmiendo en Memoria Secundaria**: Está inactivo y ha sido movido a disco para liberar espacio en memoria principal.
7. **Cambio de Contexto**: El proceso se prepara para pasar de modo supervisor a usuario, pero el kernel asigna la CPU a otro proceso.
8. **Creación (Transición)**: Estado inicial en que el proceso ha sido creado pero aún no está listo para ejecutarse.
9. **Zombie**: Estado final tras ejecutar exit(). El proceso finalizó, pero permanece con algunos datos disponibles para el proceso padre.

**Ejemplo Básico de Ciclo de Vida**

* **Inicio**: Un proceso se crea, pasando por el estado **Creación**.
* **Listo para Ejecutar**: Espera a que el planificador lo seleccione.
* **Ejecución en Modo Usuario**: Pasa a ejecutarse en modo usuario.
* **Cambio a Modo Supervisor**: Si el proceso necesita realizar una operación de sistema, pasa a modo supervisor.
* **Dormir en Memoria**: Puede quedarse inactivo (dormido) si está esperando un recurso.
* **Zombie**: Finalmente, tras la llamada exit(), el proceso se convierte en zombie, esperando que el proceso padre recoja sus datos antes de eliminarlo completamente.