**El Proceso.**

**De Programa a Proceso.**

**El programa:**

1. Programador edita código fuente
2. El compilador compila el source code en una secuencia de instrucciones máquina y datos llamada.
3. El compilador genera esa secuencia y posteriormente se guarda en disco: programa ejecutable.



**Compilación:**

**Fase de procesamiento:** El preprocesador (cpp) modifica el código de fuente original de un programa escrito en C de acuerdo a las directivas que comienzan con un carácter (#). El resultado de este proceso es otro programa en C con la extinción .i

**Fase de compilación:** el compilador (cc) traduce el programa .i a un archivo de texto .s que contiene un programa en lenguaje assembly.

**Fase de ensamblaje:** a continuación el ensamblador (as) traduce el archivo .s en instrucciones de lenguaje de maquina empaquetándolas en un formato conocido como programa objeto realocable. Este es almacenado en una archivo con extensión .o

**Fase de link edición:** generalmente los programas escritos en lenguaje C hacen un uso de funciones que forman parte de la biblioteca estándar de C que es provista por cualquier compilador de ese lenguaje. Por ejemplo la función printf(), la misma se encuentra en un archivo objeto pre compilado que tiene que ser mezclado con el programa que se está compilando, para ello el linker realiza esta tarea teniendo como resultado un archivo objeto ejecutable.

Diagrama

Descripción generada automáticamente

**Formato ejecutable**

Un programa es un archivo que posee toda la información de cómo construir un proceso en memoria [KER](cap. 6).

* Instrucciones de Lenguaje de Máquina: Almacena el código del algoritmo del programa.
* Dirección del Punto de Entrada del Programa: Identifica la dirección de la instrucción con la cual la ejecución del programa debe iniciar.
* Datos: El programa contiene valores de los datos con los cuales se deben inicializar variables, valores de constantes y de literales utilizadas en el programa.
* Símbolos y Tablas de Realocación: Describe la ubicación y los nombres de las funciones y variables de todo el programa, así como otra información que es utilizada por ejemplo para debugg.
* Bibliotecas Compartidas: describe los nombres de las bibliotecas compartidas que son utilizadas por el programa en tiempo de ejecución, así como también la ruta del linker dinámico que debe ser usado para cargar dicha biblioteca.
* Otra información: El programa contiene además otra información necesaria para terminar de construir el proceso en memoria.

**Un programa en Linux: ELF.**

Diagrama

Descripción generada automáticamente

* **readelf** is a Unix binary utility that displays information about one or more ELF files. A

free software implementation is provided by GNU Binutils.

* **elfutils** provides alternative tools to GNU Binutils purely for Linux.[11]
* **objdump** provides a wide range of information about ELF files and other object

formats. objdump uses the Binary File Descriptor library as a back-end to structure

the ELF data.

* The Unix **file** utility can display some information about ELF files, including the

instruction set architecture for which the code in a relocatable, executable, or shared

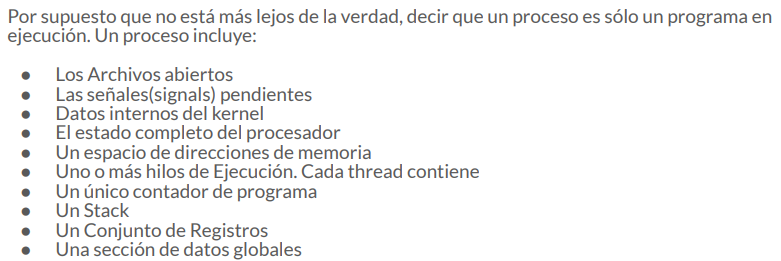
object file is intended, or on which an ELF core dump was produced

El sistema operativo más precisamente el **Kernel** se encarga de:

* Cargar instrucciones y datos de un programa ejecutable en memoria.
* Crear el Stack y el Heap.
* Transferir el Control al programa.
* Proteger al SO y al programa.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



“Un Proceso es una entidad abstracta, definida por el Kernel, en la cual los recursos del sistema son asignados” [KER]

**Virtualización:**

* Virtualización de Memoria.
* Virtualización de Procesamiento.

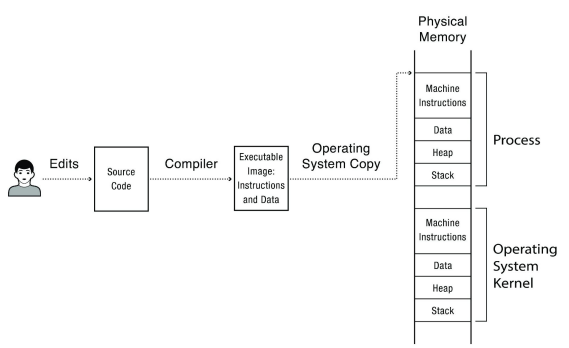


Imagen que contiene Diagrama

Descripción generada automáticamente

**Virtualización de Memoria**

* Uno de estos mecanismos es denominado Memoria Virtual, la memoria virtual es una abstracción por la cual la memoria física puede ser compartida por diversos procesos.
* Un componente clave de la memoria virtual son las direcciones virtuales, con las direcciones virtuales, para cada proceso su memoria inicia en el mismo lugar, la dirección 0.
* Cada proceso piensa que tiene toda la memoria de la computadora para sí mismo, si bien obviamente esto en la realidad no sucede. El hardware traduce la dirección virtual a una dirección física de memoria.

**Traducción de direcciones**

Se traduce una Dirección Virtual (emitida por la CPU) en una Dirección Física (la memoria). Este mapeo se realiza por hardware, más específicamente por Memory Management Unit (MMU).

**Virtualización de Procesador**

La virtualización de procesamiento es la forma de virtualización más primitiva, **consiste en dar la ilusión de la existencia de un único procesador para cualquier programa que requiera de su uso.**

El SO crea esta ilusión mediante la virtualización de la CPU a través del kernel.

Tabla

Descripción generada automáticamente

**Proceso: el contexto**

* El contexto de un proceso es la información necesaria para describir al proceso.
* Cada proceso posee un contexto.
* Según bach:” el contexto de un proceso comprende, el contenido del address

space, el contenido de los registros de hardware y las estructuras de datos que

pertenecen al kernel relacionadas con el proceso” .

* El contexto de un proceso es la unión de:
  + User-level context
  + Register context
  + System-level context

**El contexto User-level**

Consiste en las Secciones

* Text
* Data
* Stack
* Heap

**El contexto Register**

* Program Counter Register
* Processor Status Register
* Stack Pointer Register
* General Purpose Registers
  + En x86

**El contexto System-level**

* La entrada en la **Process Table Entry**
* La **u area**
* La Process Region Entry, Region Table y Page Table que definen el mapeo de la

memoria virtual vs memoria física del proceso.

* El Stack del Kernel, contiene los stack frames de las llamadas al kernel hechas por

el proceso.

Son dos estructuras que pertenecen al Kernel:

* la **process table** contiene información que siempre tiene que estar disponible para el kernel.
* La **u área** contiene campos que sólo deben estar disponibles cuando el proceso está corriendo.

**Process table:**

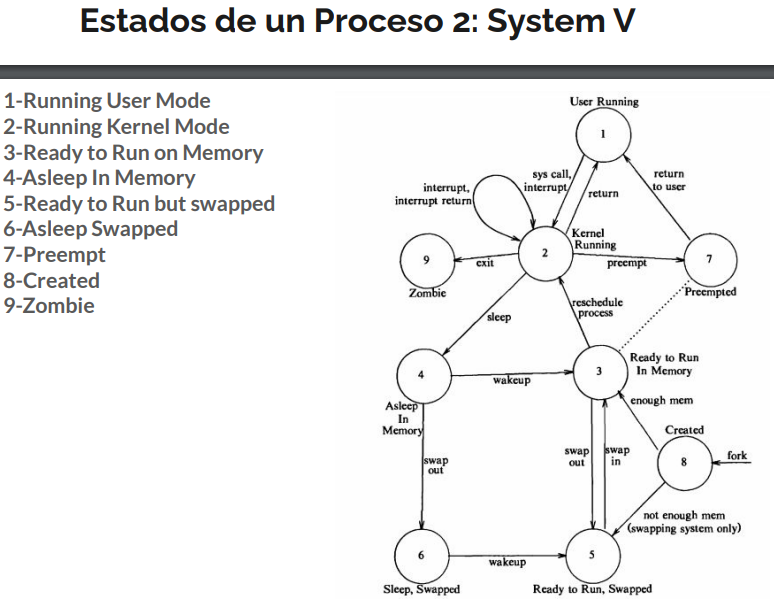
* dentificación: cada proceso tiene un identificador único o process ID (PID) y además, perteneces a un determinado grupo de procesos.
* Ubicación del mapa de direcciones del Kerner del u area del proceso.
* Estado actual del proceso
* Un puntero hacia el siguiente proceso en el planificador y al anterior.
* Prioridad
* Información para el manejo de señales.
* Información para la administración de memoria.

Contenido de la user area <arch/x86/include/asm/user.h>:

* Un puntero a la proc structure del proceso
* El UID y GID real
* Argumentos para, y valores de retorno o errores hacia, la system call actual
* Manejadores de Señales
* Información sobre las áreas de memoria text,data, stack, heap y otra información.
* La tabla de descriptores de archivos abiertos (Open File descriptor Table).
* Un puntero al directorio actual
* Datos estadísticos del uso de la cpu, información de perfilado, uso de disco y límites de recursos.

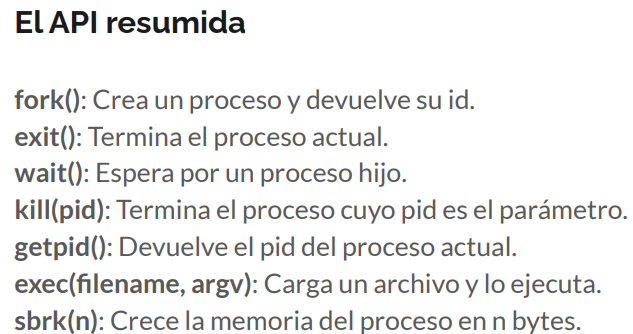
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente



Texto

Descripción generada automáticamente



Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

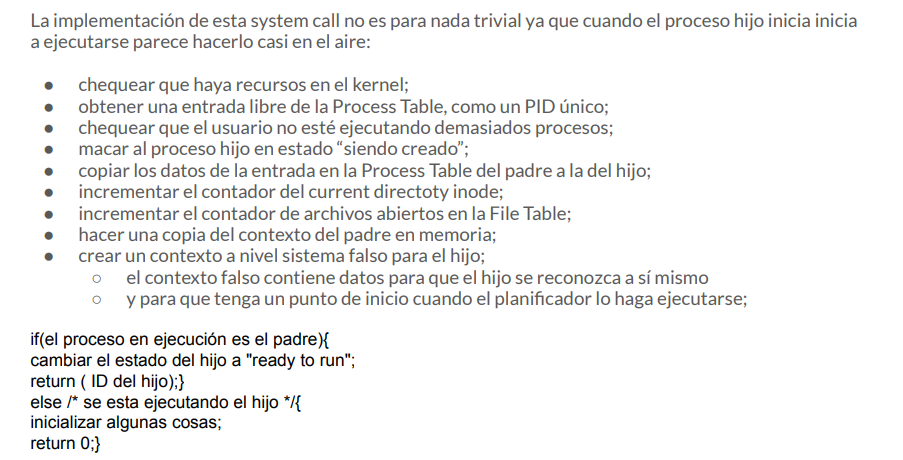


Imagen que contiene interior, pájaro, ave

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

**Procesos en Linux**

Texto

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

No hay una lista de task\_struct, sino que cada los mismos task\_struct tienen una lista que se entrelazan entre ellas 🡪 lo hace mucho Linux.

Diagrama

Descripción generada automáticamente