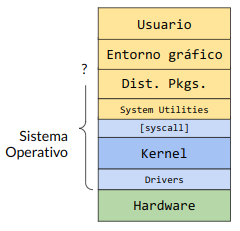
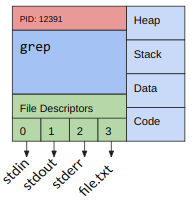
Introducción

* **Driver**: Software que interactúa con el hardware
* **Kernel**: Decide cómo administrar los recursos que tenemos. Controla y administra el hardware. Administra los drivers. Algunos drivers vienen embebidos en el kernel. No es suficiente para proveer al usuario de todas las cosas que necesita. Corre con privilegios elevados y es el núcleo del sistema operativo. Por encima de él hay distintas herramientas codeadas que proveen distintas herramientas y son parte del sistema operativo.
* **Syscall**: Interfaz del kernel. Conjunto de funciones para que el usuario pueda comunicarse con el kernel. Es un contrato. No es algo codeado sino que está especificado.
* **System Utilities**: Programas básicos que se ejecutan del lado del usuario (pocos privilegios) pero facilitan el uso final. Conjunto de utilidades, programas diferentes y chiquitos y se combinan en las utilidades del sistema. No son parte del kernel (no corren con privilegios). Podríamos decir que acá “terminaría” el sistema operativo, pero no tiene un límite definido realmente.
* **Distribution Packages**: Permiten administrar qué utilidades del sistema tenés o no.
* **Entorno Gráfico**
* **Usuario**: El que usa la computadora.
* **UNIX**: Familia de sistemas operativos. Tienen la misma interfaz para el usuario final (shell). Es una forma de implementar un sistema operativo que se asemeja al original. Tiene un kernel, una línea de comandos, librerías y headers, y utilidades del sistema (ls, ps, find, grep, sed).
* **Estándares POSIX y SUSv4**: Qué quiere decir ser un sistema tipo UNIX. Documentos que definen qué se necesita para que un sistema operativo sea de tipo UNIX. SUSv4 surgió después de POSIX.
* **Ubuntu/Fedora/etc**: Distribución de sistema operativo tipo UNIX. Toma las bases, hasta las system utilities de UNIX. Tienen la misma base pero por encima tienen otras cosas que quedan a criterio de cada distribución. En la capa del kernel tienen Linux. Por encima le meten un manejador de escritorio, de paquetes, etc.
* **Linux**: Es un kernel. Todas las distribuciones de linux tienen un kernel linux y utilidades de sistema de GNU. Se ajustan al estándar POSIX.
* **GNU**: ‘Is Not Unix’. Tiene una implementación de todas las utilidades del sistema y del kernel. La mayoría de las distribuciones de tipo UNIX tienen las utilidades de GNU => deberían llamarse distribuciones basadas en GNU/Linux.
* **Línea de Comandos**: Espera una línea de input. Hace algo y me devuelve el prompt. La línea de comandos de todos los sistemas operativos tipo UNIX, por cumplir con el estándar POSIX entienden todas las mismas instrucciones. Facilita scripting y automatización.
* **Shell**: Programa que lee los inputs, los interpreta y ejecuta y muestra salida. Ejemplo: bash. Provee de funcionalidades para poder combinar programas y hacer cosas complicadas. El pipe (‘|’) es una funcionalidad de la shell. Funciona como pegamento entre los comandos.
* **Proceso**: Instancia de un programa. Cada uno tiene su propia memoria (heap, stack, code, etc), su nombre, su identificador único y sus “archivos abiertos”. En UNIX, ps muestra la lista de procesos que está ejecutando el usuario.

Lab fork

Fecha límite de entrega: 2 semanas (09/09/2022).

Vamos a estar del lado del usuario. Codeamos utilidades del sistema. Interactuamos con el kernel a través de las syscalls. Implementamos 4 utilidades:

1. ping-pong (de fantasía)
2. primes (de fantasía)
3. find (real)
4. xarg (real)

ping-pong

Crear dos procesos: padre e hijo. El padre tiene que enviarle un mensaje al hijo (número aleatorio). El hijo tiene que decirle al padre que recibió el mensaje y enviarle ese mismo número al padre. Una vez que el hijo hace eso, termina y el padre espera a escuchar al hijo y termina. Vamos a usar la función fork.

fork(): Crea un nuevo proceso idéntico al proceso que llama a fork, la única diferencia es el valor que retorna. En un lugar devuelve 0 (significa que es el proceso hijo) y en otro lado (en el padre) devuelve el process\_id del proceso hijo.

pipe(): Una tubería. Es unidireccional: el extremo de escritura y el de lectura. Los extremos son archivos que el kernel los conecta a través de la estructura de pipe. La lectura bloquea el proceso hasta que haya algo que leer. La escritura bloquea el proceso hasta que se pueda escribir. El kernel mantiene un buffer para el caño que tiene cierto tamaño. Cuando escribo en un pipe, el kernel me va a guardar el mensaje en ese buffer y lo va llenando hasta que alguien en el otro extremo lo vaya leyendo. Se puede llenar, y cuando esto pasa la escritura se bloquea hasta que se libere espacio. Recibe un arreglo de dos posiciones: en la 0 guarda el extremo de lectura de pipe; en la 1 el extremo de escritura. Cada pipe tiene su propio buffer.

primes

Implementar el algoritmo de la criba de Eratóstenes: todos los enteros, encontramos los primos, tachamos todos los múltiplos de dos, leemos cuál es el primo más chico, tacha todos los múltiplos, leemos cuál es el más chico, tacha todos los múltiplos, etc. Así, el primer número que se encuentra sin tachar es un número primo y se puede ir listando. N procesos donde cada uno lee por un extremo los números que le pasa el proceso anterior, los filtra y se los escribe al siguiente proceso. Generamos dinámicamente N procesos comunicados por N pipes entre medio para calcularle números primos. Llamamos a primes con un número y es la cantidad de números que generamos. Esperamos que la consigna imprima todos los números primos y etc. Los números se pasan uno a uno, no mediante una lista. Hacemos un write por cada número que pasamos.

La mejor forma de implementarlo es una suerte de función recursiva donde cada vez que detectas que necesitas un nuevo proceso, en ese momento llames a fork y hagas que el hijo llame a una función que te resuelva todo eso. Lo que tiene que pasar es que al momento que detectas que necesitas un nuevo filtro, generar un pipe y hacer un for para poder empezar a escribirlos. La recursividad vive mientras que se sigan recibiendo cosas del pipe. Cuando todos los extremos de escritura del pipe se cierran, devuelve 0.