Comunicación Inter-Procesos (**I**nter-**P**rocess **C**ommunication) IPC

Marzo 2015

Comunicación Inter-Procesos

- Característica básica en los Sistemas Operativos
- El objetivo de IPC es compartir espacios de memoria entre los procesos del sistema
  - Con variables, segmentos de memoria o buffers
  - Se utilizan bibliotecas del SO especializadas para IPC
- Es necesario utilizar técnicas de comunicación y sincronización entre sí
  - Normalmente se utiliza un paso de mensajes a un nivel bajo-intermedio

Características

- La comunicación entre procesos sigue un conjunto de reglas de comunicación
- Los procesos pueden ser ejecutados en dos niveles
  - En una misma computadora
  - En varias computadores conectadas a través de una red
- Los SO proveen dos instrucciones enviar (*write*) y recibir (*read*)
  - Permiten crear enlaces unidireccionales o bidireccionales para la comunicación

Entornos de los SO

- POSIX – Interfaz de Sistema Operativo Portable
  - (**P**ortable **O**perating **S**ystem; **I**nterface) – **X** por UNIX
  - Sugerido por Stallman en la década de los 80's
- Norma de la IEEE para definir una interfaz estandarizada del SO y el entorno. Incluye:
  - Una interfaz de comandos
  - Utilerías para apoyar la portabilidad de aplicaciones
- Más información: http://pubs.opengroup.org/onlinepubs/009695399/

Partes de POSIX (1/2)

- POSIX 1. Core Services
  - Creación y control de procesos, señales, excepciones por violación de segmento, punto flotante, instrucciones ilegales
  - Errores de los buses de comunicación, temporizadores, operaciones de archivos y directorios (sobre un sistema de archivos montado)
  - Tuberías (pipes), Biblioteca C, instrucciones de entrada y salida
- POSIX 1b. Extensiones para tiempo real
  - Planificación con uso de prioridades, señales en tiempo real, temporizadores
  - Semáforos, intercambio de mensajes, memoria compartida, entrada/salida síncrona y asíncrona, y bloqueos de memoria

Partes de POSIX (2/2)

- POSIX 1c. Extensiones para hilos
  - Creación, control y limpieza de hilos
  - Planificación, sincronización y manejo de hilos
- POSIX 2. Shell y utilidades
  - Interfaz de comandos
  - Utilerías
- La versión actual es POSIX:2008

SO compatibles con POSIX

- Totalmente
  - BSD/OS, LynxOS, Solaris, VxWorks, UnixWare
- Mayoritariamente
  - BeOS, GNU/Linux, NetBSD, OpenSolaris
- POSIX en Windows
  - Cynwin desarrollo compatible para entornos de ejecución Windows
  - Windows Services for Unix permite compatibilidad con productos Microsoft

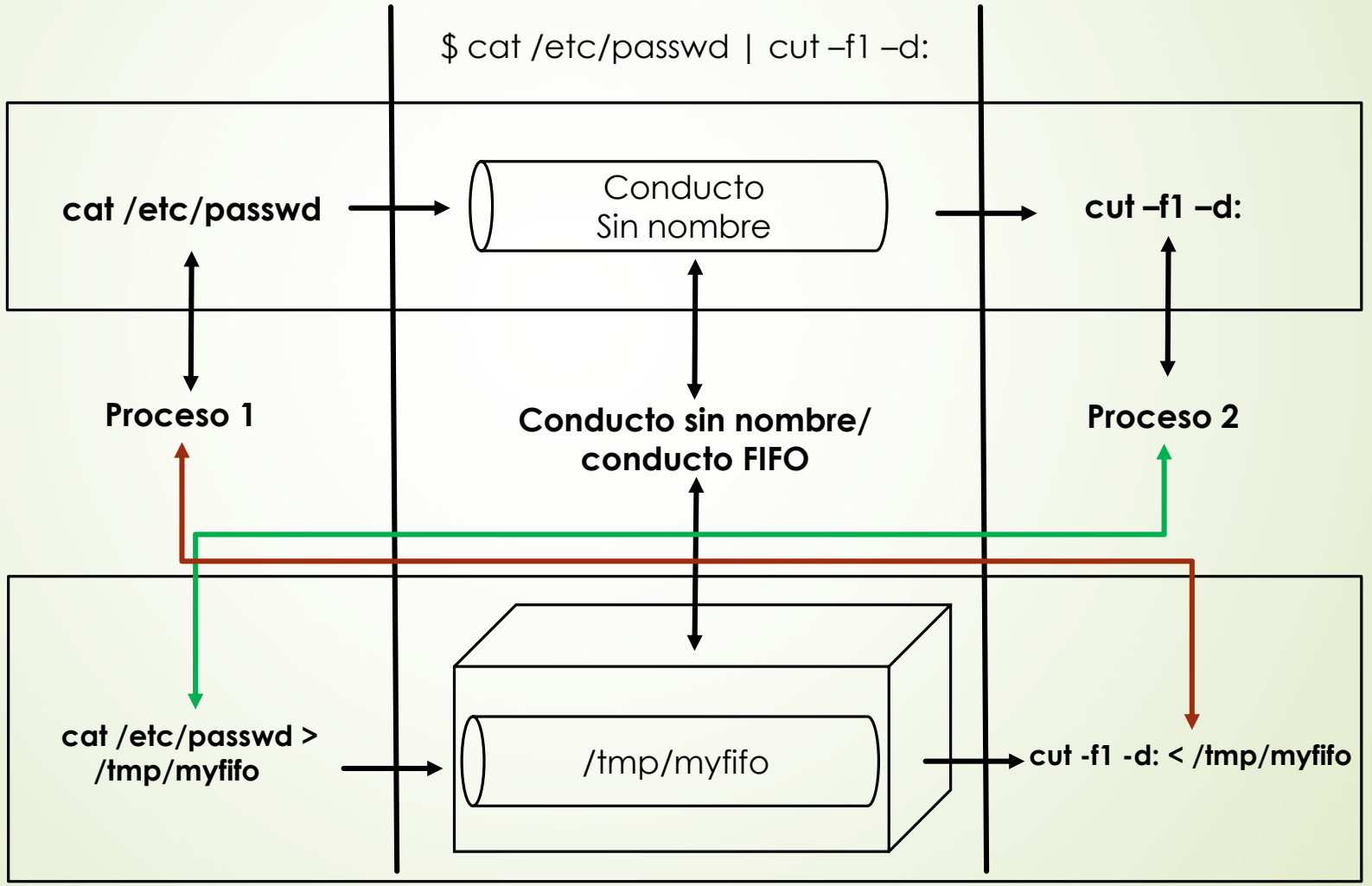
Métodos más Utilizados

Nombre	
Señales al núcleo	Local
Archivos	
Tuberías con Nombre o sin Nombre (POSIX)	
Memoria Compartida (POSIX)	
Semáforos (POSIX)	
Cola de menajes	
Puertos lógicos o físicos	Local, Remoto
Socket y paso de mensajes(RMI, CORBA)	Remoto

Tuberías o Conductos

- Existen dos tipos
  - 1) Sin nombre o anónimos
  - 2) Con nombre o FIFO
- Los conductos sin nombre no existen en el sistema de archivos
- Los conductos con nombre son del tipo FIFO (Primero en entrar primero en salir) y existen necesariamente en el sistema de archivos (también son llamados conductos FIFO)

Tuberías o Conductos



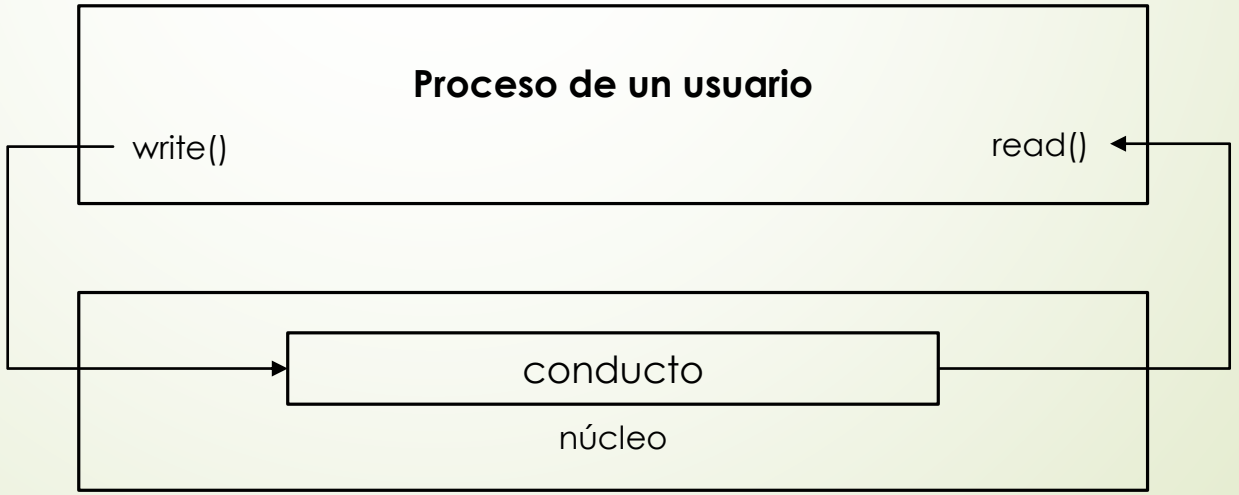
$ cut -f1 -d: < /tmp/myfifo & cat /etc/passwd > /tmp/myfifo

Tuberías o Conductos Sin Nombre

- En el escenario anterior los conductos sirven para alimentar la entrada y salida
- Otro ejemplo con un conducto sin nombre
- $cut –f1 –d: < /etc/group | sort
- Se crean procesos hijos con la llamada ***fork()***

Tuberías o Conductos Sin Nombre

- Son half-dúplex, los datos solo pueden viajar en una dirección
- Solo se utilizan entre procesos relacionados, aquéllos que tiene un proceso ancestro en común
- La lectura y escritura de los conductos es con la llamada a la función *pipe*
  - *Int pipe(int filedes[2]);*
- Si la llamada a la función tiene éxito se abren dos descriptores de archivos. El primer descriptor filedes[0] para leer y filedes [1] para escribir



Tuberías o Conductos Sin Nombre

- La norma general es que el proceso de lectura cierre el descriptor de escritura y el proceso de escritura cierre el descriptor de lectura, por ejemplo:
  - Si el proceso padre está enviando datos al hijo (escribiendo), el padre cierra filedes[0] y escribe en filedes[1], mientras que el hijo cierra filedes[1] y lee de filedes[0]
  - Si el hijo está enviando datos (escribiendo) al padre, el hijo cierra filedes[0], y escribe en filedes[1], mientras que el padre cierra filedes[1] y lee en filedes[0]

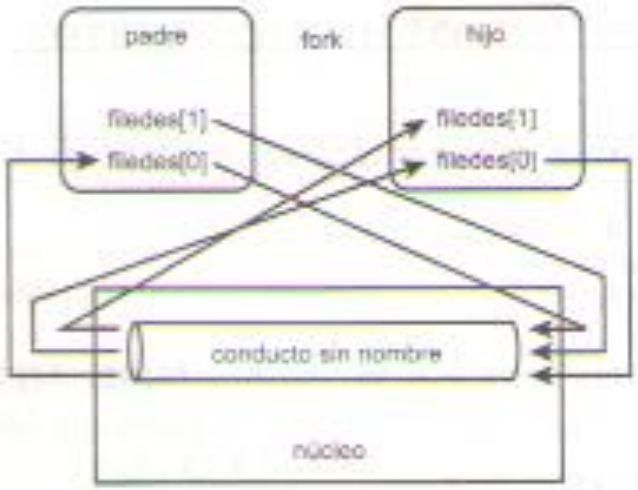


FIGURA 17.3. Lectura y escritura en un conducto después de un fork.

Tuberías FIFO

- Son persistentes
- Existen en el sistema de archivos
- Pueden ser más útiles que los conductos sin nombre porque permiten intercambiar datos de procesos no relacionados
- La función para crear una FIFO es mkfifo

- Por ejemplo:
  - $mkfifo –m 600 fifo1
  - Utilizar el programa popen.c:
    - cat < fifo1 | cut -c1 -5 &
    - ./popen > fifo1
    - #si fifo1 fuera un archivo normal el resultado sería un archivo con la salida de popen

Actividad

- Implementar y analizar los ejemplos de tuberías sin nombre y FIFO incluidos en el libro “Programación en Linux”, pág. 353-367

- Proponer y modificar los programas rdfifo.c y wrfifo.c para adaptarlos a un caso práctico, eres libre de elegir el escenario