



Programación de Servicios y Procesos

UD1 – Gestión de procesos (parte II)

Creación de Procesos en C

- Procesos en linux y otros SOs
- Tres operaciones para gestionar creación de procesos:
 - `system()`
 - `fork()`
 - `execl`

Creación de Procesos en C

- Procesos en linux y otros SOs
- Tres operaciones para gestionar creación de procesos:
 - `system()` se le pasa como parametro un comando(que puede tener parametros tambien) pasar interprete de comandos
 - `fork()` mitosis de programa crea hijos, los hijos tendran que morir antes de acabar el padre
 - `execl` le pasa una aplicacion como argumento solicita para generar un nuevo programa

Creación de Procesos en C

- system()

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
    printf("Ejemplo de uso de system():");
    printf("\n\tListado del directorio actual y envio a un fichero:");
    printf("%d",system("ls > ficsalida"));
    printf("\n\tAbrimos con el gedit el fichero...");
    printf("%d",system("gedit ficsalida"));
    printf("\n\tEste comando es erróneo: %d",system("ged"));
    printf("\nFin de programa...\n");
}
```

Creación de Procesos en C

- fork()

```
#include <stdio.h>
#include <unistd.h>

void main(void)
{
    pid_t id_pactual, id_padre;    variables de tipo pid_t

    id_pactual = getpid();
    id_padre = getppid();

    printf("PID de este proceso: %d\n", id_pactual);
    printf("PID del proceso padre: %d\n", id_padre);
}
```

Creación de Procesos en C

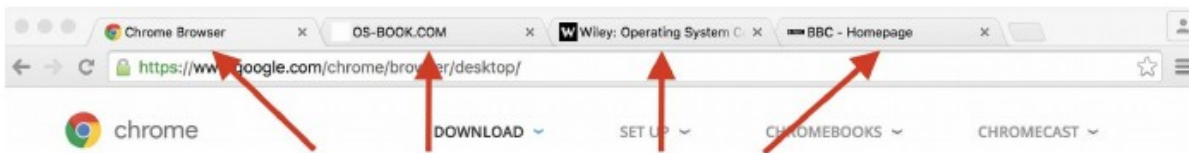
- `execl()`
- ```
#include <unistd.h>

int execl(const char *fichero, const char *arg0, ...,
 char *argn, (char *)NULL);
```



# Ejemplo de Arquitectura Multiproceso - Chrome

- Muchos navegadores web corren en un único proceso.
  - Si una web se cuelga, todo el navegador falla.
- Google Chrome tiene un sistema multiproceso con tres tipos diferentes:
  - Proceso browser maneja interfaz usuario, disco y red
  - Proceso renderer maneja las páginas web lidiando con html, javascript. Un subproceso de renderer se abre para cada navegador
  - Proceso plug-in se genera un proceso por unidad de estos.



Each tab represents a separate process.

16/10/2023

7

# Comunicación entre procesos

- Término original: Inter-Process Communication IPC
- En SO similares a linux-unix hay varias formas de comunicación entre procesos:
  - Pipes (con y sin nombre)
  - Colas de mensajes tubería "horizontal"
  - Semáforos estructura que genera un flag, permitiendo o no ejecutar el proceso
  - Segmentos de memoria compartida



# Comunicación entre procesos (II)

- Ejemplo de uso de pipes sin nombre

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main() {
 int fd[2]; // fd[0] es para lectura, fd[1] es pa
 char mensaje_padre[] = "Hola hijo!";
 char mensaje_hijo[50];

 // Crear el pipe
 if (pipe(fd) == -1) {
 perror("pipe");
 return 1;
 }

 // Crear el proceso hijo
 pid_t pid = fork();

 if (pid == -1) {
 perror("fork");
 return 1;
 }
}
```

# Comunicación entre procesos (II)

- Ejemplo de uso de pipes con nombre o fifos

```
int main() {
 int fd;
 char mensaje[100];

 // Abre el FIFO en modo lectura
 fd = open("myfifo", O_RDONLY);

 if (fd < 0) {
 perror("open");
 return 1;
 }

 // Lee del FIFO
 read(fd, mensaje, sizeof(mensaje));

 printf("Mensaje leído: %s", mensaje);

 // Cierra el descriptor de archivo
 close(fd);

 return 0;
}
```

# Comunicación entre procesos (III)

- Ejemplo de uso semáforos

```
int main() {
 key_t key = ftok("semaphore_
 int sem_id = semget(key, 1,

 if (sem_id == -1) {
 perror("semget");
 return 1;
 }
}
```

indicara si se puede usar, o no

# Entendiendo la programación concurrente

- Programación concurrente → Programación no determinista
- Condiciones de carrera → Resultado depende del orden de ejecución de los procesos o hilos → Ej: dos procesos modifican misma variable.
- Sincronización de procesos → Secciones críticas