LLamadas al Sistema Unix

Sistemas Operativos

Escuela de Ingeniería Civil Informática

Cambiar el código de un proceso
exec()

UNIVERSIDAD DEL BÍO-BÍO

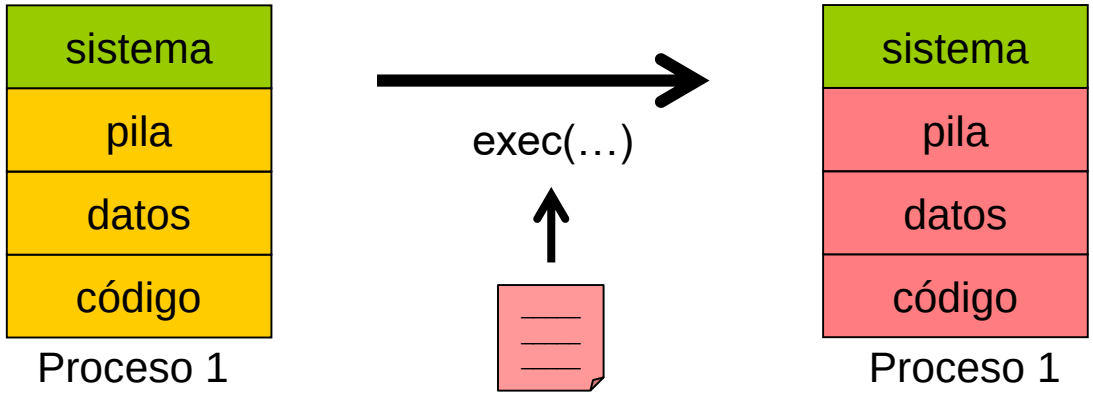
Cambiar el Código de un Proceso

- La función *exec()* permite cambiar el código de un proceso (para que no sea un clon).
- En Unix existe un conjunto de 6 llamadas al sistema, denominadas “la familia exec”.
- Las llamadas se pueden clasificar según los parámetros que utilizan:

Llamada al sistema	Formato de los argumentos	Traspasa el ambiente actual de las variables	La búsqueda del PATH es automática
execl	lista estática	Si	No
execv	arreglo dinámico	Si	No
execle	lista estática	No	No
execve	arreglo dinámico	No	No
execlp	lista estática	Si	Si
execvp	arreglo dinámico	Si	Si

¿Cómo funciona exec?

- La función *exec()* reemplaza los segmentos de código, datos y pila del proceso.
- Si es exitoso entonces *exec()* no retorna al antiguo proceso, porque se cambia la imagen del proceso.
- Si *exec()* falla, se continúa con la siguiente instrucción después de *exec()*.
- El nuevo programa comienza su ejecución en la función *main()*.



¿Cómo funciona exec?

- Tomemos como ejemplo *execlp()* que utiliza una lista estática de parámetros, traspasa las variables de ambiente y utiliza /bin como ruta de búsqueda del nuevo código (programa).

- Los parámetros típicos son:

int **execlp(const char *_file_, const char *_arg0_, ..., const char *_argn_, char * /*NULL*/**);

ruta al programa en disco

parámetros pasados al programa (igual que invocar un programa desde el intérprete de comandos

NULL para indicar fin de los parámetros, por portabilidad

- Ejemplo:

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

main() {
  execlp("cal","cal","2017",NULL);
  printf("Este texto no será mostrado si exec es exitoso.\n");
}
```

Supongamos que se desea ejecutar un programa calendario (cal).

Ejemplo 1 - execlp

- Ejecutar el comando cat de Unix

```
$ a.out  test.txt
```

```
execlp("/bin/cat",  "cat", argv[1], (char *)NULL);
```

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

int main(int argc, char *argv[ ]){
  if (argc > 1) {
    execlp("/bin/cat", "cat", argv[1], (char *)NULL);
    perror("error en exec ");
    exit(1);
  }
  printf("Modo de uso: %s archivo_texto\n", *argv);
  exit(1);
}
```

arg[0] para cat

Nombre del archivo test.txt

Formato de la función *exec()*

- Los siguientes son los prototipos de las funciones:

```
#include <unistd.h>

int execl(const char *path, const char *arg0, ..., const char *argn, char * /*NULL*/);

int execv(const char *path, char *const argv[ ]);

int execle(const char *path, const char *arg0, ..., const char *argn, char * /*NULL*/,
                char *const envp[ ]);

int execve(const char *path, char *const argv[ ], char *const envp[ ]);

int execlp(const char *file, const char *arg0, ..., const char *argn, char * /*NULL*/);

int execvp(const char *file, char *const argv[ ]);
```

Clasificación de funciones exec

- Las llamadas también se pueden clasificar en dos grupos:

1. `execl()`, `execlp()`, `execle()`

   Estas llamadas pasan los argumentos de la línea de comandos del programa mediante una lista de constantes. Son útiles cuando se conoce el n° de argumentos que se van a pasar.

   - `execle()` permite pasar nuevos valores a variables de ambiente.
   - `execlp()` permite tomar el PATH por defecto del ambiente.

2. `execv()`, `execvp()`, `execve()`

   Estas llamadas pasan los argumentos de la línea de comandos en un arreglo de argumentos (dinámico), por lo cual es útil cuando no sabemos el n° de argumentos que hay.

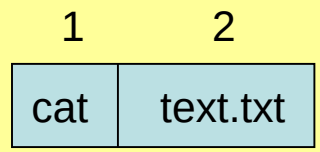
   - `execve()` permite pasar nuevos valores a variables de ambiente.
   - `execvp()` permite tomar el PATH por defecto del ambiente.

Ejemplo 2 - execvp

- Ejecutar el comando cat de Unix

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

int main(int argc, char *argv[ ]){
  execvp(argv[1], &argv[1]);
  perror("error en exec ");
  exit(1);
}
```



- Como invocar este programa

```
$ a.out   cat test.txt
```

0 1 2

```
int execvp(const char *file,
           char *const argv[ ]);
```

Ejemplo: fork y exec juntos

- En la **mayoría de los casos (99%)**, `fork()` y `exec()` son utilizados en conjunto.