

Sistemas Operativos Primer

Recuperatorio

Parte Práctica

UTN FRD Ingeniería en Sistemas de
Información Dpto. Ingeniería en Sistemas
de Información Área: Computación

Apellido y Nombres

1.- Realizar un proceso que reciba desde línea de comando una secuencia de comandos a ejecutar y que cree un hijo por cada comando a ejecutar (crea un hijo ejecuta el comando, crea otro hijo ejecuta el comando, crea otro hijo ejecuta el comando, mientras existan comandos). NO USAR system.

Los comandos no tienen modificadores. El padre espera la terminación de cada hijo. Ejemplo de la línea de comandos \$. /a.out ps ls cal who.

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
main(int argc, char * argv[]) {
    pid_t pid ; int i ; for(i = 1
    ; i < argc ; i++) {
        pid = fork() ; if
        ( pid == 0 )
    execlp(argv[i],argv[i],0); else
```

```
    wait(0); } exit(0); }2.- Dado los prototipos de las
```

siguientes funciones

```
int EscribirPidEnArchivo(int, pid_t)
    Retorna -1 en caso de error o 0 en caso de
    éxito Recibe el descriptor del archivo y el pid
```

```
pid_t LeerPidDeArchivo(int)
    Retorna -1 en caso de error o el pid en caso de
    éxito Recibe el descriptor del archivo.
```

void TareaHijo(void) (se utiliza para todos los hijos), esta tarea hace que el proceso hijo ejecute una tarea durante un tiempo determinado.

void TerminarHijo(pid_t) esta función la utiliza el padre para terminar un

hijo

El objetivo de este problema es la creación de un proceso que genere n procesos en abanico (n se ingresa como parámetro desde línea de comandos). Cada proceso hijo ejecuta la tarea TareaHijo el padre los termina en el orden en el que fueron creados, luego elimina el archivo y termina. El padre utiliza un archivo para guardar los pid de los procesos.

1 de 4 14/06/2012

**Sistemas Operativos Primer
Recuperatorio
Parte Práctica**

**UTN FRD Ingeniería en Sistemas de
Información** Dpto. Ingeniería en Sistemas
de Información Área: Computación

Apellido y Nombres

```
#include <stdio.h>
#include <stdlib.h>
#include
<sys/types.h>
#include <unistd.h>
#include <fcntl.h>
#include <string.h>
```

```
void TareaHijo(void); int
EscribirPidEnArchivo(int,long);
void TerminarHijo(long);
```

```
main(int argc,char * argv[])
{
    pid_t pid ; int i ; int fd =
    open("pids",O_CREAT|O_TRUNC|O_RDWR,0600); for(i = 0
    ; i < atoi(argv[1]) ; i++) {
        pid = fork() ; if
        ( pid == 0 )
        TareaHijo(); else
        { EscribirPidEnArchivo(fd,pid); } } char pidss[4];
```

```
lseek(fd,0,0); while(read(fd,pidss,4))
```

```
TerminarHijo(atoi(pidss)); close(fd); exit(0); }void
```

```
TareaHijo(void) { while(1) sleep(1); }
```

```
int EscribirPidEnArchivo(int fd,long pid)
```

```
{ char pidss[4]; sprintf(pidss,"%li",pid);
```

```
if (write(fd,pidss,4) > 0) return 0; else
```

```
return -1; }void TerminarHijo(long pid) {
```

```
printf("Termino Hijo %d\n",pid); kill(pid,9); }
```

2 de 4 14/06/2012

Sistemas Operativos Primer Recuperatorio

Parte Práctica

UTN FRD Ingeniería en Sistemas de Información Dpto. Ingeniería en Sistemas de Información Área: Computación

Apellido y Nombres 3.- Realizar el siguiente

proceso, Para que los procesos hijos escriban el abecedario en el archivo, el hijo1

escribe en letras mayúsculas y el hijo2 escribe en letras minúsculas, las letras no

pueden aparecer repetidas, pero no necesariamente el abecedario tiene que quedar en

orden. Prototipos para el manejo de archivos: int open(char *, int, int); int write(int,char

*, int); Hijo1 int read(int, char *, int);

```
fork()
```

```
write
```

```
Padre
```

```
Abecedario
```

```
fork()
```

```
write Hijo2 #include <stdio.h> #include <stdlib.h> #include <sys/types.h> #include
```

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

```
main() {
```

```
pid_t pid ; int letra ; int fd =
```

```
open("abecedario",O_CREAT|O_TRUNC|O_WRONLY,0600) ; pid = fork() ; if ( pid
```

```
== 0 )
```

```
{
```

```
letra = 'A'; sleep(1); while(letra <= 'Z')
```

```
{
```

```
write(fd,&letra,1); letra += 2; } } else {
```

```
pid = fork(); if (pid == 0)
```

```
{ // sleep(1);
```

```
letra = 'b'; while(letra <= 'z')
```

```
{  
write(fd,&letra,1); letra += 2; } }
```

3 de 4 14/06/2012

Sistemas Operativos Primer
Recuperatorio
Parte Práctica

UTN FRD Ingeniería en Sistemas de
Información Dpto. Ingeniería en Sistemas
de Información Área: Computación

Apellido y Nombres

```
else
```

```
{
```

```
wait(0); wait(0); close(fd); } } exit(0); }
```

4 de 4 14/06/2012