Sistemas operativos

TP 2: Llamadas al sistema

2022 – 2C

Grupo 7

Chiani Franco chianifranco58@gmail.com

Cozzi Estéfano estefanocozzi@gmail.com

Meolans Joel joemeolans2002@gmail.com

Tabla de contenido

Sección principal

Ejercicio 1. ARGUMENTOS POR LÍNEA DE COMANDO (ARGC – ARGV – OPCIONES).

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

int main(int argc, char* argv[]){
    int valor;
    if(argc==1) printf("Uso: %s [-cl] [-s nombre_archivo]\n", argv[0]);
// Si no hay parametros

    while(valor=getopt(argc, argv, "cls:")){

        if(valor==-1) break;
            switch(valor){

                case 'c': printf("Cantidad de argumentos: %i\n",
argc);  // Si se invoca con -c
                break;

                case 'l': ; // Si se invoca con -l
                    int i;
                    for(i=0;i<argc;i++){
                    printf("Argumento: %i- %s.\n", i, argv[i]);
                }
                break;

                case 's': ; // Si se invoca con -s
                    int fd;
                    fd = creat(optarg, 0600);
                    fd = open(optarg, O_WRONLY | O_APPEND | O_CREAT,
0644);
                    int j;
                    for(j=0;j<argc;j++){
                        write(fd, argv[j], strlen(argv[j]));
                    }
                    close(fd);
                break;
            }
        }
}
```

Ejercicio 2. GESTIÓN DE PROCESOS.

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

int main(int argc,char *argv[]){

    int valor = atoi(argv[1]);
    printf("NODO R - VALOR = %d\n", valor);
    printf("ID proceso raiz: %d\n" , getpid());
    int nodo2;
    int nodo1 = fork();

    if(nodo1==0){
        int nodo12;
        printf("NODO 1 - VALOR = %d\n", valor+100);
        printf("ID NODO 1: %d - ID padre NODO 1(NODO R): %d\n", getpid(),
getppid());

        int nodo11 = fork();
            if(nodo11==0){
                printf("NODO 1.1 - VALOR = %d\n", (valor+100)*2);
                printf("ID NODO 1.1: %d - ID padre NODO 1.1(NODO 1):
%d\n", getpid(), getppid());
            } else { nodo12 = fork(); }

            if(nodo12==0){
                printf("NODO 1.2 - VALOR = %d\n", (valor+100)/2);
                printf("ID NODO 1.2: %d - ID padre NODO 1.2(NODO 1):
%d\n", getpid(), getppid());
            }

    } else { nodo2 = fork(); }

    if(nodo2==0){
        printf("NODO 2 - VALOR = %d\n", valor-100);
        printf("ID NODO 2: %d - ID padre NODO 2(NODO R): %d\n", getpid(),
getppid());
    };

return 0;
}
```

Ejercicio 3. GESTIÓN DE PROCESOS.

```c
// ej3.c
#include <stdio.h>
#include <unistd.h>
int main(int argc,char *argv[]){

    printf("Id proceso que invoca: %d\n", getpid());

    if (argc==1){ printf("Error. Usar ./Nombre_pgr_invoca [numero]\n"); }
    else {

        char* arg_list[] = {argv[1], NULL};
        execv("./factorial", arg_list);
    }
return -1;
}
```

```c
// factorial.c
#include <stdio.h>
#include <unistd.h>

int factorial(int val);

int main(int argc,char *argv[]){
    int valor = atoi(argv[0]);
    if (valor>=1 && valor<=10){
        printf("Factorial de %d = %d\n", valor, factorial(valor));
    } else {
        printf("Numero fuera de rango: %d\n", valor);
    }
    printf("Id de Prg-Factorial: %d\n", getpid());
return 0;
}

int factorial(int val){
    if(val==0){ return 1; }
    else if (val==1) { return 1; }
    else { return val * (factorial (val-1)); }
}
```

La función execv() reemplaza el proceso que se está ejecutando por uno nuevo que es llamado en el argumento de la función. Al no iniciarse un nuevo proceso, sino que se superpone al que ya se estaba ejecutando, el PID no se ve afectado y no cambia durante la ejecución.

Por lo que podemos ver que al ejecutar el ej3, ambos id serán idénticos.

Ejercicio 4. GESTIÓN DE SEÑALES.

```c
// tostadora.c
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <unistd.h>

#define NUM_HUESPEDES 30

int cantTostadas = 0;

void tostar(int tipoPan);

int main(int argc, char * argv[]){

    signal(SIGUSR1, &tostar);
    signal(SIGUSR2, &tostar);

    while(cantTostadas != NUM_HUESPEDES);

    return 0;
}

void tostar(int tipoPan){
    cantTostadas+=1;
    switch(tipoPan){
        case SIGUSR1:
            printf("Deseo tostadas de pan blanco. \n");
            printf("Tostadas de pan blanco. (%i) \n", cantTostadas);
            sleep(2);
            break;
        case SIGUSR2:
            printf("Deseo tostadas de pan negro. \n");
            printf("Tostadas de pan negro. (%i) \n", cantTostadas);
            sleep(1);
            break;
    }

    printf("Tostadas listas. \n");
    printf("Tostadora libre, esperando pan. \n");

}
```

```c
// huesped.c
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
int pid_c, i;
pid_c = atoi(argv[1]);
for(i=1; i<=30; i++){
if(i%5!=0){
printf("Pedido de tostada de Pan Blanco. Huesped: %d\n", i);
kill(pid_c, SIGUSR1);
}
else{
printf("Pedido de tostada de Pan Negro. Huesped: %d\n", i);
kill(pid_c, SIGUSR2);
}
sleep(3);
}
return 0;
}
```

Ejercicio 5. GESTIÓN DE SEÑALES.

```c
// controlc.c
#include <stdio.h>
#include <signal.h>

int main(){
    int i;
    signal(SIGINT, SIG_IGN);
    printf("Iteracion INICIADA. Presionar Ctrl-C NO tiene efecto....\n");
    for(i=1;i<=100000;i++){
        if(i%1000==0){
            printf("Iteracion %i\n", i);
            sleep(1);
        }
    }
    printf("Computation is done.\n\n");
    signal(SIGINT, SIG_DFL);
    printf("REINICIO de la Iteracion. Presionar Ctrl-C AHORA tiene
efecto....\n");

    for(i=1;i<=100000;i++){
        if(i%1000==0){
```

```
            printf("Iteracion %i \n", i);
            sleep(1); }}}
```

Ejercicio 6. INFORMACION DEL SISTEMA.

```c
// infosistema.c
#include <sys/utsname.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char * argv[]){

    struct utsname unameData;
    uname(&unameData);

    printf("Tipo de Sistema: %s\n", unameData.sysname);
    printf("Nombre del Equipo: %s\n", unameData.nodename);
    printf("Version del Kernel: %s\n", unameData.release);
    printf("Version del S.O.: %s\n", unameData.version);
    printf("Arquitectura: %s\n", unameData.machine);

    return 0;
}
```

Sección de descargas