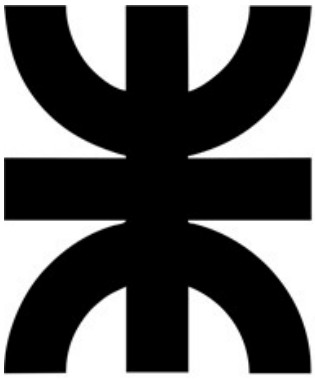
Trabajo práctico nro 4: "Sistema de Archivos y Administración de Memoria "



Universidad Tecnológica Nacional Facultad Regional Santa Fe

Asignatura: Sistemas Operativos

Segundo cuatrimestre, año 2019

Identificación del grupo: 13

Nombres de los integrantes del grupo:
Cabaña, Juan Pablo
Dalmasso, Fidel José
Kloster, Narella Katherine

Direcciones de correo electrónico:
juanpablocaban98@outlook.com
fideldalmasso@gmail.com
narellakloster79@gmail.com

ACLARACIÓN:

El proyecto se divide en 4 archivos fuentes ejercicio1.c ejercicio2.c ejercicio3.c y buddyFS.c; y una cabecera: buddyFS.h.

Para compilar el proyecto, por favor, ejecutar el comando "make" desde la terminal, dentro de la carpeta donde se encuentran estos archivos.

El filesystem se leera siempre desde la ruta /*dev*/fd0.

BuddyFS: SISTEMA DE ARCHIVOS y SISTEMA COMPAÑERO (buddy algorithm)

buddyFS.c

```
#include "buddyFS.h"
#include <fcntl.h>
#include <getopt.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <linux/fs.h>
#include <linux/ext2_fs.h>
// si linux/ext2_fs.h no existe, instalar e2fslibs-dev
// y utilizar la libreria siguiente
//#include <ext2fs/ext2_fs.h>
#include <errno.h>


void mostrarError(char texto[]) {
    printf("Error numero: %d\n", errno);
    perror(texto);
    exit(1);
}

void mostrarUso(){
        printf("Modo de uso\nbuddyFS [-s][-l][-b tamanioNube]\nEl archivo se leera siempre en la ruta /
dev/fd0\n");
     exit(0);
       }

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

    int fd1, posTablaInodos,c,tamBloque,entradaDirectorio, cantEntradasDirectorios;
    __le32  cantInodos, idBloqueDeEntradasRaiz;

    struct ext2_super_block * sb = malloc(sizeof(struct ext2_super_block));
       struct ext2_group_desc * gd = malloc(1024);

    if (argc < 2)
               mostrarUso();

    if ((fd1 = open("/dev/fd0", O_RDONLY) ) == -1) mostrarError("open");

       //superbloque
       cargarSuperbloque(fd1,sb);

       //groupDescriptor
       cargarGroupDescriptor(fd1,gd);
```

```
       //tabla de inodos
    cantInodos = (sb->s_inodes_count) - (sb->s_free_inodes_count);
    posTablaInodos = (gd->bg_inode_table * 1024);
       struct ext2_inode *ti = (struct ext2_inode *)malloc(cantInodos * sizeof(struct ext2_inode));
       cargarTablaInodos(fd1,ti,posTablaInodos,cantInodos);

       //tablaEntradasDirectorios
    tamBloque = sb->s_log_block_size + 1024;
    idBloqueDeEntradasRaiz = ti[1].i_block[0];
    entradaDirectorio = idBloqueDeEntradasRaiz * tamBloque;

    struct ext2_dir_entry_2 *tde = (struct ext2_dir_entry_2  *)malloc(cantInodos * sizeof(struct
ext2_dir_entry_2 ));
    cantEntradasDirectorios = cargarTablaEntradasDirectorios(fd1,tde, entradaDirectorio,tamBloque);



    if ((c = getopt (argc, argv, "slb")) < 0) mostrarError("Debe ingresar un argumento");

    switch (c) {
    case 's':
        ejercicio1(*sb);
        break;
    case 'l':
        ejercicio2(fd1, *sb, *gd, ti, tde, cantEntradasDirectorios);
        break;
    case 'b':
        if(argc != 3){
            printf("Debe ingresarse el tamanio de la nube, y solo uno.\n");
        } else {
    int tamNube = atoi(argv[2]);
            ejercicio3(ti, tde, cantEntradasDirectorios, tamNube);
        }
        break;
    default:
               mostrarUso();
    }

    return 0;

}
```

-s lectura de información del superbloque

Ejercicio1

ejercicio1.c

```
#include "buddyFS.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <linux/magic.h>
#include <linux/fs.h>
#include <linux/ext2_fs.h>
// si linux/ext2_fs.h no existe, instalar e2fslibs-dev
// y utilizar la libreria siguiente
//#include <ext2fs/ext2_fs.h>
#include <errno.h>
#include "buddyFS.h"

#define EXT2_S_IFMT     0xF000  /* format mask  */
#define EXT2_S_IFLNK    0xA000  /* symbolic link */
#define EXT2_S_IFREG    0x8000  /* regular file */
#define EXT2_S_IFDIR    0x4000  /* directory */


void cargarSuperbloque(int fd,  struct ext2_super_block * sb){

       if (lseek(fd, 1024, SEEK_SET) == -1) mostrarError("lseek");
       if (read(fd, sb, 1024) == -1) mostrarError("read");

       if (sb->s_magic != EXT2_SUPER_MAGIC) {
              printf("%s\n", "No hay sistema de archivos tipo EXT2");
              exit(1);
       }
}

void ejercicio1(struct ext2_super_block sb) {


       printf("-----------------INFORMACION DEL SB-----------------\n");

       printf("%-40s%12s\n", "Sistema de archivos tipo:", "EXT2");
       printf("%-40s%12s\n", "Nombre del volumen:", sb.s_volume_name);
       printf("%-40s%12d\n", "Cantidad de i-nodos:", sb.s_inodes_count);
       printf("%-40s%12d\n", "Cantidad de i-nodos libres:", sb.s_free_inodes_count);
       printf("%-40s%12d\n", "Primer i-nodo no reservado:", sb.s_first_ino);
       printf("%-40s%12d\n", "Tamanio estructura de un i-nodo:", sb.s_inode_size);
       printf("%-40s%12d\n", "Tamanio de bloque:", sb.s_log_block_size + 1024);
       printf("%-40s%12d\n", "Primer bloque de datos:", sb.s_first_data_block);
       printf("%-40s%12d\n", "Cantidad de bloques:", sb.s_blocks_count);
       printf("%-40s%12d\n", "Cantidad de bloques libres:", sb.s_free_blocks_count);
       printf("%-40s%12d\n", "Tamanio total en disco:", sb.s_blocks_count * (sb.s_log_block_size +
1024));
       printf("----------------------------------------------------\n");

}
```

-l listado de archivos

Ejercicio2

ejercicio2.c

```
#include "buddyFS.h"
#include <fcntl.h> //open
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <time.h>
#include <grp.h>
#include <pwd.h>
#include <linux/fs.h>
#include <linux/ext2_fs.h>
// si linux/ext2_fs.h no existe, instalar e2fslibs-dev
// y utilizar la libreria siguiente
//#include <ext2fs/ext2_fs.h>


#define EXT2_S_IFMT     0xF000  /* format mask  */
#define EXT2_S_IFLNK    0xA000  /* symbolic link */
#define EXT2_S_IFREG    0x8000  /* regular file */
#define EXT2_S_IFDIR    0x4000  /* directory */


void fechaF(int segundos, char fechaS []);
void grupoF(int gid, char grupoS []);
void obtenerModo(int modo, char salida[11]);
void usuarioF(int uid, char usuarioS []);

void cargarGroupDescriptor(int fd, struct ext2_group_desc * gd){
       if (lseek(fd, 1024 * 2, SEEK_SET) == -1) mostrarError("lseek");
       if (read(fd, gd, 1024) == -1) mostrarError("read");

}


void cargarTablaInodos(int fd1,struct ext2_inode *ti, int posTablaInodos,int cantInodos){

       struct ext2_inode * i = (struct ext2_inode*)malloc(sizeof(struct ext2_inode));
       int p;
       for (p = 0; p < cantInodos; p++) {
              if (lseek(fd1, posTablaInodos + (p * 128), SEEK_SET) == -1) mostrarError("lseek");
              if (read(fd1, i, 128) == -1) mostrarError("read");
              ti[p] = *i;
       }
}

int cargarTablaEntradasDirectorios(int fd1,struct ext2_dir_entry_2  *tde,int entradaDirectorio,int
tamBloque){

       struct ext2_dir_entry_2 * de = malloc(sizeof(struct ext2_dir_entry_2));

       int n=0;
       while (1) {
              if (lseek(fd1, entradaDirectorio, SEEK_SET) == -1) mostrarError("lseek");
              if (read(fd1, de, sizeof(struct ext2_dir_entry_2)) == -1) mostrarError("read");

              tde[n]=*de;
```

```
                tde[n].name[tde[n].name_len] = '\0';

                entradaDirectorio += de->rec_len;
                n++;
                if ((entradaDirectorio % tamBloque) == 0) break;

        }
        return n;

}




void fechaF (int segundos, char fechaS []) {
        time_t fechaAux = segundos;
        struct tm *fechaAux2 = localtime(&fechaAux);
        strftime(fechaS, 80, "%b %d %Y", fechaAux2);
}

void grupoF (int gid, char grupoS []) {
        struct group* grupoAux = getgrgid(gid);
        strcpy(grupoS, grupoAux->gr_name);
}

void usuarioF(int uid, char usuarioS []) {
        struct passwd* usuarioAux = getpwuid(uid);
        strcpy(usuarioS, usuarioAux->pw_name);
}

void obtenerModo(int modo, char salida[11]) {

        char salida2[11] = "---------- ";
        char salida3[11] = "-rwxrwxrwx ";

        int aux = EXT2_S_IFMT & modo; //aplicar mask

        switch (aux) {
        case EXT2_S_IFDIR:
                salida2[0] = 'd';
                break;
        case EXT2_S_IFREG:
                salida2[0] = '-';
                break;
        case EXT2_S_IFLNK:
                salida2[0] = 'l';
                break;
        }

        int i;
        for (i = 0; i < 9; i++) {
                //los permisos estan ubicados en los primeros 9 bits
                if ((modo >> i) & 1)
                        salida2[9 - i] = salida3[9 - i];
        }

        salida2[10] = '\0';
        strcpy(salida, salida2);

}

void ejercicio2(int fd1, struct ext2_super_block sb, struct ext2_group_desc gd, struct ext2_inode ti[],
struct ext2_dir_entry_2 tde[], int cantEntradasDirectorios) {


        printf("%-8s %-16s %-6s %-8s %-8s %-8s %-16s %-16s\n", "Inodo", "Modo", "Links", "Usr", "Grp",
"Tamanio", "Fecha", "Archivos");
        int i = 0;
```

```
	for(i=0;i<cantEntradasDirectorios;i++) {
		char usuarioTexto[16];
		char grupoTexto[16];
		char fechaTexto[24];
		char modoTexto[11];

		obtenerModo(ti[tde[i].inode - 1].i_mode, modoTexto);
		fechaF(ti[tde[i].inode - 1].i_mtime, fechaTexto);
		grupoF(ti[tde[i].inode - 1].i_gid, grupoTexto);
		usuarioF(ti[tde[i].inode - 1].i_uid, usuarioTexto);

		int cantLinks = ti[tde[i].inode - 1].i_links_count;
		printf("%-8d %-16s %-6d %-8s %-8s %-8d %-16s %-16s\n", tde[i].inode, modoTexto,
cantLinks, usuarioTexto, grupoTexto, ti[tde[i].inode - 1].i_size, fechaTexto, tde[i].name);

	}

}
```

-b sistema compañero (buddy)

Ejercicio3

ejercicio3.c

```
#include "buddyFS.h"
#include <stdio.h>
#include <stdlib.h>

int asignar2(int tamNube,int tamArchivo, int arbol[]);
void asignar(int tamNube,int tamArchivo, int arbol[],int falla[1]);
int espacioSuficiente(struct ext2_inode ti [], struct ext2_dir_entry_2 tde [], int
cantEntradasDirectorios, int tamNube);
void imprimir(int tamNube,int nodo, int arbol[]);
void particionar(int nodo, int arbol[]);
int place(int nodo, int arbol[]);
int potencia(int base,int exp);

void particionar(int nodo, int arbol[]){
	while(nodo!=0){
		nodo=nodo%2==0?(nodo-1)/2:nodo/2;
		arbol[nodo]=1;
	}
}

int place(int nodo, int arbol[]){
	while(nodo!=0){
		nodo=nodo%2==0?(nodo-1)/2:nodo/2;
		if(arbol[nodo]>1)
			return 0;
	}
	return 1;
}

int potencia(int base,int exp){
	int i,resp;
	if(exp==0) return 1;
	resp=base;
	for(i=1;i<exp;i++)
		resp*=base;
	return resp;
}

void imprimir(int tamNube,int nodo, int arbol[]){
	int permission=0,llimit,ulimit,tab;

	if(nodo==0)
		permission=1;
	else if(nodo%2==0)
		permission=arbol[(nodo-1)/2]==1?1:0;
	else
		permission=arbol[nodo/2]==1?1:0;

	if(permission){
		llimit=ulimit=tab=0;

		while(1){
			if(nodo>=llimit && nodo<=ulimit){
				break;
			}else{
				tab++;
				printf(".");
```

```
                                llimit=ulimit+1;
                                ulimit=2*ulimit+2;
                        }
                }

                printf(" %d ",tamNube/potencia(2,tab));

                if(arbol[nodo]>1)
                        printf("---> Asignado %db\n",arbol[nodo]);
                else
                        if(arbol[nodo]==1)
                        printf("------> \n");
                        else
                                printf("|---> Libre\n");

                imprimir(tamNube,2*nodo+1,arbol);
                imprimir(tamNube,2*nodo+2,arbol);
        }
}

void asignar(int tamNube,int tamArchivo, int arbol[], int falla[1]){
        int nivelActual=0, tamActual=tamNube, i=0;

        if(tamArchivo>tamNube){
                printf("Asignacion de %db FALLIDA (no hay mas espacio)\n",tamArchivo);
                falla[0]=1;
                return;
        }

        while(1){
                if(tamArchivo<=tamActual && tamArchivo>(tamActual/2)){
                        break;
                }else{
                        tamActual/=2;
                        nivelActual++;
                }
        }

        for(i=potencia(2,nivelActual)-1;i<=(potencia(2,nivelActual+1)-2);i++){
                if(arbol[i]==0 && place(i,arbol)){
                        arbol[i]=tamArchivo;
                        particionar(i,arbol);
                        printf("Asignacion de %db EXITOSA\n",tamArchivo);
                        break;
                }
        }

        if(i==potencia(2,nivelActual+1)-1){
                printf("Asignacion de %db FALLIDA (no hay mas espacio)\n",tamArchivo);
                falla[0]=1;
        }
}

int asignar2(int tamNube,int tamArchivo, int arbol[]){
        int nivelActual=0, tamActual=tamNube, i=0;

        if(tamArchivo>tamNube){
                return 0;
        }

        while(1){
                if(tamArchivo<=tamActual && tamArchivo>(tamActual/2)){
                        break;
                }else{
                        tamActual/=2;
                        nivelActual++;
                }
        }
```

```
	for(i=potencia(2,nivelActual)-1;i<=(potencia(2,nivelActual+1)-2);i++){
		if(arbol[i]==0 && place(i,arbol)){
			arbol[i]=tamArchivo;
			particionar(i,arbol);
			return 1;
		}
	}

	if(i==potencia(2,nivelActual+1)-1){
		return 0;
	}

	return -1;
}

int espacioSuficiente (struct ext2_inode ti [], struct ext2_dir_entry_2 tde [], int
cantEntradasDirectorios, int tamNube){
	int i=0;
	int *pesoArchivos=(int*) malloc (cantEntradasDirectorios*sizeof(int));
	int *arbol=(int*) malloc (tamNube*sizeof(int));

	for (i=0; i<cantEntradasDirectorios; i++){
		pesoArchivos[i]=ti[tde[i].inode-1].i_size;
		if (asignar2(tamNube,pesoArchivos[i],arbol)==0){
			return tamNube=espacioSuficiente(ti,tde,cantEntradasDirectorios,tamNube*2);
		}
	}
	return tamNube;
}

void ejercicio3 (struct ext2_inode ti [], struct ext2_dir_entry_2 tde [], int cantEntradasDirectorios,
int tamNube){
	int i=0, espacioOcupado=0;
	int *pesoArchivos=(int*) malloc (cantEntradasDirectorios*sizeof(int));
	int *arbol=(int*) malloc (tamNube*sizeof(int));
	int falla[1]={0};
	for (i=0; i<cantEntradasDirectorios; i++){
		pesoArchivos[i]=ti[tde[i].inode-1].i_size;
		espacioOcupado+=pesoArchivos[i];
		asignar(tamNube,pesoArchivos[i],arbol,falla);
		imprimir(tamNube,0,arbol);
		printf("\n");
	}


	if(falla[0]==1){
		printf("ESPACIO NECESARIO PARA REALIZAR TODAS LAS ASIGNACIONES: %d\n",
espacioSuficiente(ti,tde,cantEntradasDirectorios,tamNube));
	}
	printf("ESPACIO TOTAL OCUPADO %db\n",espacioOcupado);

}
```

Descargos

Ejercicio 1

- No hay descargos.

Ejercicio 2

- No hay descargos a realizar.

Ejercicio 3

- Asumimos que al no poder alojar un archivo en la nube (cuando se produce un fallo) se debe intentar cargar el resto de los archivos de todas formas.
- Asumimos que cuando se pide “determinar el tamaño de nube que permite realizar todas las asignaciones de archivo“ se solicita que en la salida del programa se incluya un apartado (“ESPACIO NECESARIO PARA REALIZAR TODAS LAS ASIGNACIONES“) con dicho tamaño de nube.

Bibliografía

1. Poirier, D. (2001-2002). *The Second Extended File System*. Eindhoven University of Technology. Recuperado de: https://www.win.tue.nl/~aeb/linux/fs/ext2/ext2.html#I-UID
2. C library function – strftime(). *Tutorialspont.* Recuperado de: https://www.tutorialspoint.com/c_standard_library/c_function_strftime.htm
3. grp.h – group structure. *The Open Group*. Recuperado de: https://pubs.opengroup.org/onlinepubs/009695399/basedefs/grp.h.html
4. getgrgid, getgrgid_r - get group database entry for a group ID. *The Open Group*. Recuperado de: https://pubs.opengroup.org/onlinepubs/009695399/functions/getgrgid.html
5. pwd.h - password structure. *The Open Group*. Recuperado de: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pwd.h.html
6. getpwuid, getpwuid_r - search user database for a user ID. *The Open Group*. Recuperado de: https://pubs.opengroup.org/onlinepubs/9699919799/functions/getpwuid.html
7. BUDDY SYSTEM. *Sources Code World.com*. Recuperado de: http://www.sourcecodesworld.com/source/show.asp?ScriptID=1077
8. Inside Ext2/3 File System. Sourceforge. Recuperado de: http://ext2read.sourceforge.net/old/ext2fs.htm
9. (2011). What is a bitmasking?. *Stackoverflow.* Recuperado de: https://stackoverflow.com/questions/10493411/what-is-bit-masking
10. Analyzing a filesystem. *Santa Monica College*. Recuperado de: http://homepage.smc.edu/morgan_david/cs40/analyze-ext2.htm
11. C program to get nth bit of a number. *Codeforwin*. Recuperado de: https://codeforwin.org/2016/01/c-program-to-get-value-of-nth-bit-of-number.html
12. Online c function prototype header generator tool. *Tinkerings of an Engineer*. Recuperado de:http://blog.olkie.com/2013/11/05/online-c-function-prototype-header-generator-tool/
13. How to declare a structure in a header that is to be used by multiple files in c?. *Stackoverflow.* Recuperado de: https://stackoverflow.com/questions/228684/how-to-declare-a-structure-in-a-header-that-is-to-be-used-by-multiple-files-in-c

14. Dynamics arrays in C. *Emory College Of Arts and Sciences.* Recuperado de: *http://www.mathcs.emory.edu/~cheung/Courses/255/Syllabus/2-C-adv-data/dyn-array.html*
15. Dynamic Memory Allocation in C using malloc(), calloc(), free() and realloc(). *GeeksforGeeks.* Recuperado de: https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/
16. https://github.com/exscape/exscapeOS/blob/master/src/include/kernel/ext2.h