SOA - FISIERE

EX.1

```
/*sa se afiseze continutul din fisier intre pozitiile m si n citite*/
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>

main () {
	int fd; char *pbuff;
	int m, n,lfis;
	char *numeFis = (char*)malloc(200);
	printf("m:");
	scanf("%d",&m);
	printf("n:");
	scanf("%d",&n);
	printf("numeFis:");
	scanf("%s", numeFis);
	// deschidere fisier f1.txt
	fd=open(numeFis,O_RDWR);
	if(fd == -1)
		perror("Eroare deschidere fisier");
	// calculare lungime fisier, prin deplasare pointer la sfarsit fisier
	//lfis=lseek(fd,0,SEEK_END);
	//pozitionare pointer pe octetul de adresa m
	lseek(fd, m, SEEK_SET);
	//alocare buffer pentru citire fisier
	pbuff=malloc(n-m);
	read(fd,pbuff,n-m);

	write(1,pbuff,n-m); //NU se face printf cu %s ptr ca pbuf
	// nu este un string
}
```

EX2

```
/*
Sa se scrie un program care creaza un fisier de 500 de octeti pe disc. In fisier se vor scrie ciclic literele alfabetului, fiecare litera de cate 16 ori. Varianta: lungimea fisierului este data de utilizator in linia de comanda
*/
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <assert.h>

int main(int argc, char** argv){
	int fd,i,j=0;
	char *p, *numeFis;
	if (argc < 3){
		perror("insuficienti parametri");
		exit(0);
	}
	j=atoi(*(argv+1));
	numeFis = *(argv+2);
	fd = open(numeFis, O_WRONLY | O_CREAT);
	if(fd==-1) perror("eroare la deschidere fisier");
	int k;
	p = malloc(1);
	for (i=0; i<j/16; i++)
	{
		*p = (char) ('a'+i%26);
		for(k=0; k<16; k++)
			write(fd, p, 1);
	}

}
```

EX4

/*Sa se scrie un program in limbajul C, numit infofis , care afiseaza informatii despre un fisier : lungimea fisierului, numele posesorului numele grupului si drepturile de acces. Afisarea se va face astfel : Numele fisierului : f1.txt Lungimea fisierului : 35 octeti Posesor : c08ab Grup : c08 Drepturi : Posesor – citire, scriere,
executie Grup – citire executie Altii – citire Indicatie: caracteristicile fisierului se pot afla cu functia stat sau prin
executia (interna, in program !) a comenzii ls , astfel: exelp(“ls”, “ls”, “-l”, “f1.txt”, “>f2.txt”, 0)).*/

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
/**
* intoarce
* 0 - nu sunt erori
* 1 - eroare la apelul functiei stat
* 2 - nu s-a putut citi numele fisierului
*/
int main(int argc, char** argv){
	int test;
	struct stat info;
	char *numeFis = (char*)malloc(200);
	/*nume fisier*/
	printf("Introduceti numele fisierului:");
	test=scanf("%s",numeFis);

	printf("Lungimea fisierului : %d octeti\n",info.st_size);
	printf("Posesor : %x\n",info.st_uid);
	printf("Grup : %x\n",info.st_uid);
	printf("Drepturi : \n");
	printf("\tPosesor â€“ %s %s %s\n",((info.st_mode & S_IRUSR)?"citire":""),
	((info.st_mode & S_IWUSR)?"scriere":""),
	((info.st_mode & S_IXUSR)?"executie":""));
	printf("\tGrup â€“ %s %s %s\n",((info.st_mode & S_IRGRP)?"citire":""),
	((info.st_mode & S_IWGRP)?"scriere":""),
	((info.st_mode & S_IXGRP)?"executie":""));
	printf("\tAltii â€“ %s %s %s\n",((info.st_mode & S_IROTH)?"citire":""),
	((info.st_mode & S_IWOTH)?"scriere":""),
	((info.st_mode & S_IXOTH)?"executie":""));
}
```

EX5

/*Sa se scrie un program in limbajul C , numit dump , care afiseaza in hexazecimal continutul unui fisier, octet
cu octet. Afisarea continutului se face pe linei astfel : - fiecare linie contine 16 octeti ai fisierului - la inceputul
fiecarei linii afisate se scrie (in hexa) adresa primului dintre cei 16 octeti de pe linie - afisarea va contine o ultima
linie suplimentara, in care se afla adresa ultimului octet al fisierului. Numele fisierului al carui continut este afisat
va fi dat de utilizator.*/

```
#include <fcntl.h> //pt O_RDONLY
#include <unistd.h>//pt lseek

int main(int argc, char* argv[]){
	//deschidere fisier
	int fd = open(argv[1], O_RDONLY);
	if(fd == -1){
		perror("eroare deschidere fisier");
		exit(0);
	}

	//preluam continutul fisierului
	long lungime = lseek(fd, 0, SEEK_END);
	lseek(fd,0,SEEK_SET);
	char* pbuff = (char*)malloc(lungime);

	read(fd, pbuff, lungime);//copierea continutului in buffer
	int k=0; int i;
	for(i=0; i<lungime; i++){
		k++;
		if(k==1){
			printf("\n%08x ",pbuff[i]);  //afiseaza adresa caracterului
		}
		if(k==16){
			k=0; //dupa 16 octeti reseteaza
		}
		printf("%02x ",pbuff[i]);    //afiseaza urmatorul caracter in hexa

	}
	printf("\n%08x ",pbuff[lungime-1]);
	printf("\nlungime %d",lungime);

	return 0;
}
```

EX6

```
/*Sa se scrie un program in limbajul C , numit texto , care : − afiseaza continutul unui fisier text − cauta un sir
de caractere in cadrul acelui unui fisier text.*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>//pt O_RDONLY
#include <unistd.h>//pt lseek
#include <errno.h>//pt perror

int main(){
	printf("introduceti numele fisierului:");
	char* numeFis = (char*)malloc(200);
	scanf("%s",numeFis);
	int fd = open(numeFis, O_RDONLY);
	if(fd == -1){
		perror("eroare la deschiderea fisierului");
		exit(0);
	}
	long lungime = lseek(fd,0,SEEK_END);
	lseek(fd,0,SEEK_SET);
	char* pbuff = (char*)malloc(lungime);
	read(fd,pbuff, lungime);
	printf("%s",pbuff);//afisarea continutului fisierului
	//cautare sir
	printf("introduceti sirul: ");
	char sir[100];
	scanf("%s",sir);
	char* linieFisier;
	int startLinie=0;
	for(int i=0; i<lungime; i++){
		if(pbuff[i]=='\n' && i-startLinie>0){
			linieFisier=(char*)malloc(500);

			strncpy(linieFisier, &pbuff[startLinie],i-startLinie);
			//copiaza in linieFisier continutul de la startlinie pana la i-start

			linieFisier[i-startLinie]='\0';//marcator final sir
			startLinie = i+1;
			if(strstr(linieFisier,sir)!=NULL){
				//daca linia curenta contine sirul
				printf("%s\n",linieFisier);
			}
		}
	}
	return 0;
}
```

EX7

```
/*Sa se scrie in limbajul C un program numit split , care decupeaza continutul unui fisier text in n_fis fisiere a
caror lungime este de l_fis octeti.*/

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

int main(int argc, char* argv[]){
	int fd, lungime;
	char* pbuff;
	char* numeOriginal; char* fisierCurent;
	int nrFisiere, lungimeFisiere;

	if(argc<4){
		perror("Eroare, insuficienti parametri");
		exit(0);
	}
	numeOriginal = argv[1];
	nrFisiere = atoi(argv[2]);
	lungimeFisiere = atoi(argv[3]);

	fd=open(numeOriginal, O_RDONLY);
	if(fd==-1){
		perror("Eroare la deschidere");
		exit(0);
	}
	lungime = lseek(fd, 0, SEEK_END);
	pbuff = (char*)malloc(lungime);
	read(fd, pbuff, lungime);
	lseek(fd, 0, SEEK_SET);
	//--
	int fd2;
	char* nr;
	int f;
	for(int i=0; i<nrFisiere; i++){
		fisierCurent = (char*)malloc(500);
		sprintf(fisierCurent,"fisier%d.txt",i);//scrie in variabila fisierCurent un nume pt acel fisier

		fd2=open(fisierCurent, O_RDWR|O_CREAT);
		write(fd2,pbuff+i*lungimeFisiere, lungimeFisiere);
	}
	return 0;
}
```

PARTE

/*Sa se scrie in limbajul C un program numit parte , care afiseaza pe ecran o parte dintr-un fisier text. Lungimea partii afisate este de l_part octeti, iar afisarea se va face incepand din pozitia p_poz*/

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

int main(int argc, char* argv[]){
	int fisier, lungime;
	char *buffer;
	int l,poz;
	if(argc<4){
		perror("Eroare, insuficienti parametri");
		exit(0);
	}
	fisier= open(argv[1],O_RDONLY);
	lungime = lseek(fisier,0,SEEK_END);
	lseek(fisier,0,SEEK_SET);
	buffer=(char*)malloc(lungime);
	if(read(fisier,buffer,lungime)!=lungime){
		perror("Eroare la citire");
		exit(0);
	}
	l=atoi(argv[2]);
	poz=atoi(argv[3]);

	write(1,buffer+poz,l);
	return 0;
}
```

COPYTAIL

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>//lseek, write
#include <string.h>//sprintf
int main(int argc, char* argv[]){
	int nr, fisier, fisnou;
	char* nume;
	char* buffer;
	int lungime;

	nume=argv[1];
	nr=atoi(argv[2]);
	fisier = open(nume,O_RDONLY);
	lungime = lseek(fisier, 0, SEEK_END);
	lseek(fisier, 0, SEEK_SET);
	buffer = (char*)malloc(lungime);

	if(read(fisier,buffer,lungime)==-1){
		perror("Eroare la citire");
		exit(0);
	}

	if(nr>lungime){
		perror("nr de octeti nu poate fi mai mare decat lungimea");
		exit(0);
	}

	char *numenou;
	sprintf(numenou,"%s%s",nume,".tail");
	fisnou=open(numenou,O_RDWR|O_CREAT);
	write(fisnou, buffer+lungime-1-nr,nr);

	return 0;
}
```

WORDCOUNT

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

#include <fcntl.h>//pt O_RDONLY
#include <unistd.h>//pt lseek
#include <errno.h>//pt perror

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

int fisier = open(argv[1], O_RDONLY);
if(fisier == -1){
	perror("eroare la deschiderea fisierului");
	return 1;
}

long lungime = lseek(fisier,0,SEEK_END);
lseek(fisier,0,SEEK_SET);
char* buffer = (char*)malloc(lungime);
if(buffer==NULL){
	perror("eroare la alocare");
	return 1;
}

size_t rezultat = read(fisier,buffer, lungime);
if(rezultat != lungime){
	perror("eroare la citirea din fisier");
	return 1;
}

//nr caractere
printf("%d ",lungime);

//nr cuvinte, linii
int linii=0;
int cuvinte=0;

char* linieFisier;
int startLinie=0;
for(int i=0; i<lungime; i++){
	if(buffer[i]=='\n' && i-startLinie>0){
		linii++;

		linieFisier=(char*)malloc(500);
		strncpy(linieFisier, &buffer[startLinie],i-startLinie);
		//copiaza in linieFisier continutul de la startlinie pana la i-start
		linieFisier[i-startLinie]='\0';//marcator final sir
```

```
//                      startLinie = i+1;
            for(int j=0; j<i-startLinie; j++){
                                if(strchr(" \t\n",linieFisier[j])==NULL &&
                                        strchr(" \t\n",linieFisier[j+1])!=NULL){
                                cuvinte++;
                                }
                        }//strchr cauta in sirul dat un caracter dat
                        startLinie = i+1;
                }
}

printf("%d %d",cuvinte,linii);

return 0;

}
```

SOA – PROCESE

EX1

/*
Sa se scrie in limbajul C un program care creeaza filiatia de procese ilustrata in figura de mai jos.
Fiecare proces va afisa un text cu numele sau precum si pid-ul propriu si cel al parintelui.
Sa se transforme in ZOMBI cele 2 procese fiu.*/

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
main () {
        int p1, p2;
        p1=fork() ;
        if(p1==0) {
                printf("\n Fiu1 are id=%d Tata are id =%d\n",getpid(),getppid()) ;
                exit(0);
        }
        p2=fork();
        if(p2==0) {
                printf("\n Fiu2 are id=%d Tata are id =%d\n",getpid(),getppid()) ;
                exit(0);
        }
        printf("\n Tata are id =%d",getpid()) ;
        sleep(10);
}
```

EX2

/* Sa se scrie in limbajul C un program care creeaza filiatia de procese ilustrata in figura de mai jos. Fiecare proces va afisa un text cu numele sau (parinte, fiu, nepot, etc), precum si pid-ul propriu si cel al parintelui.
Sa se transforme in ORFANI procesele nepot.*/

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
main () {
        int p1, p2, p3, p4 ; int m=5, n=100;
        p1=fork() ;
        if(p1==0) {
                printf("\n Fiu1 are id=%d Tata are id =%d\n",getpid(),getppid()) ;
                p3=fork();
                if(p3==0) {
                        printf("\n Nepot1 are id=%d Tatal lui are id =%d\n",getpid(),getppid()) ;
                        sleep(20);
                }
                sleep(10);
                exit(0);
        }
        p2=fork();
        if(p2==0) {
                printf("\n Fiu 2 are id=%d Tata are id =%d\n",getpid(),getppid()) ;
                p4=fork();
                if(p4==0) {
                        printf("\n Nepot2 are id=%d Tatal lui are id =%d\n",getpid(),getppid()) ;
                        sleep(20);

                }
                sleep(10);
                exit(0);
        }

        printf("\n Tata are id =%d\n",getpid(),getppid()) ;
        sleep(30);
        exit(0);
}
```

EX3

```
/*
Sa se scrie in limbajul C  un program care creeaza filiatia de procese
 ilustrata in figura de mai jos. [parinte-fiu-nepot]
Procesul nepot1 va trimite prin pipe-line 500 de octeti ,
, procesul parinte ii va afisa pe ecran.
*/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

//parinte-fiu-nepot
int main(){
	int fiu, nepot, p[2];
	pipe(p);
	fiu=fork();
	if(fiu==0){
		printf("fiu are pid=%d, parintele are pid=%d",getpid(),getppid());

		nepot=fork();
		if(nepot==0){
			printf("nepot are pid=%d, parintele are pid=%d",
				getpid(),getppid());
			close(p[0]);
			int k=0;
			for(int i=0; i<10; i++)	{
				write(p[1],"A",1);
				sleep(1);
			}
			exit(0);
		}

		exit(0);
	}

	close(p[1]);
	char* c;
	c = (char*)malloc(1);
	while(read(p[0], c, 1)==1){
		write(1,c,1);
		sleep(1);
	}

	return 0;
}
```

PARENT

/*Să se realizeze în limbajul C un program numit parent0 care să creeze trei procese (inclusiv părintele). Fiecare proces își afișează PID-ul, primul proces fiu numără de la 1- 50000, al doilea proces fiu de la 50000-100000, iar procesul părinte așteaptă terminarea fiecăruia dintre cele două procese, afișează câte un mesajapoi se încheie la rândul sau*/

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h> //pt strcat
//parinte-fiu-nepot
int main(){
	int pid, fiu2, p[2], status=0;
	pipe(p);
	pid=fork();
	if(pid==0){
		printf("[f pid=%d, p pid=%d]",getpid(),getppid());
		close(p[0]);
		int k=0;
		char* s=(char*)malloc(100);
		char* aux=(char*)malloc(10);
		for(int i=0; i<10; i++) {
			sprintf(aux, "%d ",i);
			strcat(s,aux);
		}
		write(p[1],s,21);
		sleep(1);
		status=1;
		exit(0);

	} else {
		fiu2=fork();
		if(fiu2==0){
			printf("[f2 pid=%d, p pid=%d]",getpid(),getppid());
			close(p[0]);
			int k=0;
			char* s=(char*)malloc(100);
			char* aux=(char*)malloc(10);
			for(int i=10; i<20; i++){
				sprintf(aux, "%d ", i);
				strcat(s,aux);//concateneaza la s sirul aux
			}
			write(p[1],s,30);
			sleep(1);
			exit(0);
		}
	}
```

```
        close(p[1]);
        char* c;
        c=(char*)malloc(10);
        while(read(p[0], c, 1)==1){
        write(1,c,1);
        sleep(1);
        }

        return 0;
}
```