Subiect dat la 234 SO 28.01.2012

1. Signatura apelului sistem fork

```
#include <unistd.h>
pid_t fork(void);
```

//pag 81

2. Numiti cele trei tipuri de IPC SYSTEM V

1. Cozi de mesaje 1. Message Queues
2. Semafoare 2. Semaphore Sets
3. Memorie Partajata 3. Shared Memory Management

//pag 102

3. Numiti algoritmul folosit pentru palanificarea proceselor folosit de sistemele de operare din familia UNIX

SCHEDULING ALGORITHM: Planificarea Round-Robin (Round-Robin scheduling)

//curs Draguluici soc7

4.Care este teoria matematica ce sta la baza proiectarii algoritmilor de detectare a deadlock-urilor?

GRAFURILE!!Teoria transmiterii informaţiei, de detectare şi corectare a erorilor sau "teorema a seriabilităţii"

O.S. runs a deadlock detection algorithm to detect the circular wait condition present for a set of processes.It uses a marking method to mark all processes that are not currently deadlocked.

5. Scrieti o functie C int *myFunct(char **name, int n, int mod) care deschide cate un fisier cu numele dat de fiecare element din nume (care are n elemente), cu caracteristicile date de mod si drepturi de acces implicite, si intoarce un tabel de n elemente continand descriptorii fisierelor deschise. In caz de eroare se va intoarce NULL.

```
int *myFunct(char **name, int n, int mod){
 int i, *Desc=(int*)malloc(n*sizeof(int));
  for(i=0;i<n;i++)  {
  if((Desc[i]=open(name[i], mod))==-1)
   {
    perror("open");
    return NULL;
   }
  }
  return Desc;
}
```

6. Scrieti o functie C int myFunct(int semid, (void *)f()) care apeleaza functia data de f in mod semaforizat de semaforul de tip mutex cu indicatorul intern dat de semid.In caz de eroare se intoarce -1 si 0 in caz de succes.

```
int myFunct(int semid, (void *)f())
{
#define KEY 1499
 semid=semget(KEY, 1, 0666 | IPC_CREAT);
 if(semid<0)
 {
  fprintf(stderr, "Eroare la obtinerea semaforului.\n");
   return -1;
  }
if(semctl(semid, 0, GETVAL)==1)
  return 0;
 union semun{
 int val;
 struct semid_ds *buf;
 ushort *array;
}arg;
 arg.val=0;
 if(semctl(id, 0, SETVAL, arg)<0) {  fprintf(stderr, "Eroare la setarea semaforului.\n");
  return -1;
 }
 (void)(*f)();
  //echivalent cu f();
 return 0;}
```

2010-2011

- Grupa 242:

1. **Viteza de transmisie în reţea este de 1Mb/s. În cât timp se vor transfera 10Tb?**
   1Tb=1024Gb; 1Gb=1024Mb => 1Tb= 1048576Mb =>10Tb=10485760Mb => Se vor transfera 10Tb in 10485760 secunde=174762 ore si 40 de secunde=7281zile 18ore si 40secunde
2. **Numiţi 5 tipuri de fişiere UNIX.**
   Fisiere de tip obijnuit,director,tub,special corespunzator perifericelor fizice(bloc sau caracter),socket,legaturi simbolice.
3. **Enunţaţi 5 caracteristici memorate de un i-nod UNIX.**
   Caracteristici memorate de un i-nod UNIX: proprietarul,drepturile de acces, adresele unde se gasesc datele,
4. **Enunţaţi 5 tipuri de periferice**.
   Periferice:Tastaura,Monitor,Imprimanta,Mouse,Scanner,Boxe. De intrare,iesire,interne,externe

5. **Se consideră definiţia de mai jos. Scrieţi o funcţie C MYSTR *myFunct(char *dir, int *n) care construieşte un vector de structuri `MYSTR` ce conţin fiecare numele fişierelor şi caracteristicile i-nodurilor din directorul cu numele `dir`. `n` comtinre (sic) numărul de elemente din `vec`. Se întoarce `NULL` în caz de eroare. (in `n` se va afla numarul de fisiere. Tu trebuie sa il aflii, nu iti este dat.)**

```
typedef struct
{
    char name[NAME_MAX];
    struct stat st;
} MYSTR;
```

```
#include <dirent.h>
// Pentru operatii pe directoare cum ar fi:* opendir()* readdir()* closedir()
#include <limits.h>// Pentru constantele NAME_MAX si PATH_MAX
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>// Pentru functia si structura 'stat'.
typedef struct
{
    char name[NAME_MAX];
    struct stat st;
} MYSTR;
MYSTR *myFunct(const char *dir, int *n)
{// Deschidem directorul aflat la calea specificata de 'dir'.
    DIR *dirp = opendir(dir);
    if(dirp == NULL)
    {   (*n) = 0;
        return NULL;
    }// Citim primul fisier din director pentru a verificadaca directorul a
//fost deschis cum trebuie si se pot executa operatiile de citire.
    struct dirent *dirs = readdir(dirp);
    if(dirs == NULL)
    {   (*n) = 0;
        closedir(dirp);
        return NULL;
    }// Setam numarul de fisiere cu 1 pentru ca citisem adineaori primul
//fisier, apoi citim restul de fisiere si incrementam n-ul de fiecare data.
// Acest pas este important pentru a sti lungimea finala a vectorului de
structuri 'MYSTR'.
    (*n) = 1;
    while( (dirs = readdir(dirp)) != NULL)
    {
        (*n)++;
    }
```

```
    closedir(dirp);// Inchidem si redeschidem directorul dat pentru a o lua
//de la capat cu citirea datelor. De data asta vom citi si numele fisierelor
//+ informatiile oferite de 'stat'.
    dirp = opendir(dir);
    if(dirp == NULL)
    {
        return NULL;
    }// Alocam vectorul de structuri mai sus mentionat.
    MYSTR *files = (MYSTR *) malloc(sizeof(MYSTR) * (*n));
    if(files == NULL)
    {
        closedir(dirp);
        return NULL;
    }// Pentru a obtine informatii la nivel de system despre un anumit
//fisier, avem nevoie de calea lui exacta.Astfel, vom aloca un vector de
//caractere (aka string) de lungime PATH_MAX (lungimea maxima a unei cai
//oarecare acceptata pe un sistem de operare POSIX).
    char *current_file_path = malloc(sizeof(char) * PATH_MAX);
    int i = 0;// Incepem citirea riguroasa a fisierelor.
// Reamintesc ca la fiecare readdir(dirp) se obtine o referinta catre un alt
//fisier din directorul referit de 'dirp'.Cand nu mai sunt fisiere, 'readdir'
//returneaza 'NULL'.
    while( (dirs = readdir(dirp)) != NULL)
    {// Luam de la fiecare fisier numele si il retinem in vectorul nostru de
structuri.
        strcpy(files[i].name, dirs->d_name);
// Aici creem calea la fisierul curent, relativa la directorul 'dir'.
// Spre exemplu, daca 'dir' = "NewFolder" si denumirea fisierului curent ar
//fi "NewFile",calea relativa ar fi"NewFolder/NewFile".
        strcpy(current_file_path, dir);
        strcat(current_file_path, "/");
        strcat(current_file_path, files[i].name);
        // Executam 'stat()' si punem informatiile in vectorul nostru.
        if( stat(current_file_path, &files[i].st) < 0)
        {
            free(current_file_path);
            free(files);
            closedir(dirp);
            return NULL;
        }
        i++;
    }// Eliberam memoria alocata de noi care nu mai foloseste programului si
//inchidem directoarele.
    free(current_file_path);
    closedir(dirp);
```

```
    return files;
}
```

6. Scrieţi o funcţie C double myFunct(char *f, double *vec, int n, int *err) unde `f` este numele unui fişier ce conţine în format binar elemente de tip double, iar `n` numărul de elemente din `vec`. Funcţia returnează produsul scalar dintre `vec` şi vectorul format din primele `n` elemente din `f`. Dacă sunt mai puţin de `n` elemente în `f`, atunci elementele lipsă vor fi considerate ca având valoarea `0.0`. `err` este parametru de ieşire şi va avea valoarea `0` în caz de success şi `-1` în caz de eroare.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
double myFunct(char *f, double *vec, int n, int *err)
{
    int fd = open(f, O_RDONLY);
    if(fd < 0)
    {   *err = -1;
        return 0.0;}
    int bytes_read;
    int i=0;
    double prod_scalar=0;
    double buf;
    while((bytes_read=read(fd,&buf,sizeof(double)))>=sizeof(double)&&i<n){
       if(bytes_read == -1){   close(fd);
                               *err = -1;
                               return 0.0;}
        prod_scalar += buf * vec[i];
        i++;}
    *err = 0;
    close(fd);
    return prod_scalar;}
```

- **Grupa 231:**

1. **Numiti doua extensii din sistemul de operare Windows care lanseaza programe (orice extensie). Sa se numeasca si tipul programelor pe care le lanseaza (ex: .bat, .exe, .ini,**

.txt, .doc, etc.)

Extensii: .cpp->Dev C++;Visual Studio C++ .doc->Microsoft Word

2. **Numiti doua periferice externe folosite pentru stocarea datelor (HDD, Flash driver, CD-ROM, disc RAM, banda magnetica, etc.)**

   CD-ROM,HDD,Stick

3. **De cate discete de m MB este nevoie pentru a stoca o informatie de dimensiune n GB?**

   1Gb=1024Mb => nGB=n*1024Mb =>Nr de dischete este (n*1024)/m +1 (daca impartirea nu este exacta)

4. **Numiti doua caracteristici de selectie pe care se bazeaza programele de schedule-ing pentru procese (prioritate, timp CPU, date fixate de utilizator prin comanda $at, etc.)**

   Caracteristici: TOATE SISTEMELE (1) ECHITATE (FAIRNESS): fiecare proces primeste o capacitate echitabila de utilizare a procesorului;(2) RESPECTAREA POLITICILOR (POLICY ENFORCEMENT): politica declarata trebuie respectata; (3) ECHILIBRU (BALANCE): mentinerea ocupata a tuturor partilor sistemului;SISTEME DE PRELUCRARE PE LOTURI: (1) PRODUCTIVITATE (THROUGHPUT): maximizarea numarului de sarcini pe ora;(2) TIMP DE RASPUNS (TURNAROUND TIME): minimizarea timpului intre introducerea sarcinii si terminarea ei; (3) GRADUL DE UTILIZARE A PROCESORULUI (CPU UTILIZATION): mentinerea procesorului ocupat tot timpul;

5. **Scrieti o functie C sub forma int myFunct(int *tab, int n), unde tab este o lista de identificatori a n procese fiu. Functia asteapta terminarea tuturor proceselor din tabela. Daca cel putin un proces nu s-a terminat, atunci se returneaza -1. Daca toate procesele s-au terminat, se returneaza media aritmetica a codurilor de retur.**

```c
#include <stdlib.h>
#include <unistd.h>
int myFunct(int *tab, int n)
{   int i;
    int suma_exit_statusurilor=0;//Luam pe rand toate pid-urile din vector
    for(i=0; i<n; i++)
    {//wait_ret=retine valoarea returnata de un apel al functiei waitpid()
    int wait_ret;//status=variabila in care se retin informatii legate de
//apelul waitpid()
       int status;//Facem un waitpid() neblocant (WNOHANG) asupra PID-ului
//curent (tab[i])
        if( (wait_ret = waitpid(tab[i], &status, WNOHANG)) < 0){
//In cerinta nu scria ce sa facem in cazul in care waitpid()esueaza am
//presupus ca functia ar returna, si in acest caz, -1.
         return -1;     }
```

```
//In cazul in care, waitpid() returneaza 0, atunci inseamna ca procesul
//curent (tab[i]) nu si-a terminat executia.Deci, putem returna -1 conform
//cerintei.
        if(wait_ret == 0){
                        return -1;  }
//Daca waitpid() a returnat o valoare strict pozitiva,atunci luam valoare
//de retur a procesului fiu si o adunam la suma.
        suma_exit_statusurilor += WIFEXITED(status); }
//Returnam media aritmetica a valorilor de retur a proceselor fiu,avand
//PID-urile in vectorul 'tab'.
    return suma_exit_statusurilor / n;}
```

6. **Scrieti o functie C sub forma int myFunct(int id, (int *)f(), int *ret) unde id e identificatorul unui semafor, care apeleaza semaforizat functia desemnata de f (inainte de apel semaforul se decrementeaza si dupa apel semaforul se incrementeaza). La adresa desemnata de ret se va depune codul de returl al functiei f. Se lucreaza doar cu primul semafor din vector. myFunct returneaza 0 in caz de succes sau -1 in caz de eroare.**

```
#include <sys/sem.h>
//Din nou, header-ul functiei dat in cerinta este gresit.Daca as fi pus
'(int *) f()' in loc de 'int (f)()' mi-ar fi dat eroare la compilare.
//Presupun cÄ header-ul e asa intrucat am dedus in acelasi mod header-ul
de la problema 4.
   int myFunct(int id, int (f)(), int *ret){
//Pentru a executa operatiunile de incrementare/decrementare trebuie sa
//definim o structura de tip 'sembuf' in care sa detaliem ceea ce vrem sa
//facem.
    struct sembuf sb;
//Alegem primul (si,in cazul nostru,singurul)semafor din array-ul de
//semafoare.
    sb.sem_num = 0;
//Selectam tipul operatiunii(Numar negativ=scade valoarea semaforului cu
acea valoare)(Numar pozitiv=creste valoarea semaforului cu acea valoare)
//(Zero=functia va intra intr-o stare de asteptare pana cand valoarea
//semaforului va fi zero).
    sb.sem_op = -1;
//Setam lista de flaguri,in cazul nostru ramanand vida.
    sb.sem_flg = 0;
//Executam operatia de decrementare.
    if(semop(id, &sb, 1) < 0)
```

```
        return -1;
//Apelam functia data ca parametru si memoram valoarea in locul indicat de
'ret'.
     *ret = f();//Acum va trebui sa facem o incrementare. Cum am definit
//adineaori care semafor si ce flaguri avem nevoie, nu va mai trebui decat
//sa modificam 'sem_op' sa incrementeze.
    sb.sem_op = 1;
//Executarea operatiei de incrementare.
    if(semop(id, &sb, 1) < 0)
           return -1;
           return 0;}
```

2009-2010

- **Grupa 232:**

1. **Numiti cele 3 categorii de virusi.**
   Virusi:Clasici,viermi,bacterii.
2. **Numiti tipurile de IPC SYSTEM V.**
   Tipuri de IPC SYSTEM V: Corzi de mesaje; Vectori de semafoare;Segmente de memorie partajata
3. **Care este situatia cea mai frecventa de esec a unui apel din familia exec?**
   O val=-1 a codului de retur primit de procesul tata indica esecul o cauza fiind umplerea tabelei de porcese a sistemului
4. **Enuntati signatura a 2 apeluri de sistem UNIX.**
   Apeluri UNIX:
5. **Scrieti o functie C char *myFunct (char *fis, int *n) care intoarce adresa unei zone de memorie unde s-au depus toate vocalele din fisierul cu numele fis. La adresa memorata de n se va depune lungimea zonei de memorie returnate. In caz de eroare se va returna NULL.**

6. **Scrieti o functie C int myFunct (int *tab, int n, (void *)funct(int x), int val) care lanseaza un proces fiu care mai intai mascheaza semnalele din vectorul tab, ce are lungime n si apoi va executa functia desemnata de funct cu argumentul val. Se va returna -1 in caz de eroare sau daca procesul fiu nu s-a terminat normal si 0 altfel.**

```c
int myFunct(int *tab, int n, void (funct) (int x), int val){
//Se creeaza un proces fiu.Din acest moment exista doua procese, ambele
//executand aceleasi instructiuni pana la al doilea 'if'.
    pid_t child = fork();
//Se verifica daca fork-ul s-a realizat cu succes.In cazul in care a
//esuat, inseamna ca un proces nou nu a mai fost creat, iar functia
//returneaza -1
    if(child < 0) {
        return -1;    }
//Aici se verifica daca procesul este fiul.Daca da,se executa
//instructiunile explicate in cerinta.Daca child este 0, atunci inseamna
//ca ne aflam in fiu.
     if(child == 0)  {
//Declaram o "multime de semnale".Aceasta este un fel de vector in care
//sunt retinute semnalele.Peste un 'sigset_t' se executa operatii folosind
//anumite functii
        sigset_t sset;
//Golim multimea de semnale.
       if( sigemptyset(&sset) < 0)
        exit(1);
        int i;
//Adaugam pe rand toate elementele din `tab` in aceasta multime
       for(i=0; i<n; i++)
            if( sigaddset(&sset, tab[i]) < 0)
                exit(1);
//Acum, peste toata multimea, punem o masca de ignorare.Altfel spus, toate
//semnalele din 'sset' vor fi ignorate de acum in colo.
    if( sigprocmask(SIG_SETMASK, &sset, NULL) < 0)
          exit(1);
//Apelam functia din cerinta.
     funct(val);
      exit(0);    }
//Aici suntem inapoi in procesul tata.Ultimul exit(0) ne asigura ca
//procesul fiu nu v-a mai ajunge vreodata sa execute instructiunile care
//urmeaza.Facem un 'wait' pana cand procesul fiu se termina de executat,
//dupa care ii vom lua codul de retur si,intr-un final vom returna cum
//spune cerinta.
       int status;
       wait(&status);
      if(WIFEXITED(status) == 0)
     return 0;
     return -1;}
```

- **Grupa 233:**
    1. **Descrieti cele 2 operatii elementare asupra semafoarelor (conform Dijkstra)**
    2. **Signatura msgsnd si msgrcv.**
    3. **Enumerati starile unui proces (nu neaparat Unix)**
    4. **Ramura matematicii care sta la baza proiectarii algoritmilor de detectare a deadlockului. (Raspuns: grafurile)**
    5. **int myFunct(int* pid, int*sig, int n) unde tablourile pis si sig au fiecare n elem. Fctia transmite fiecarui proces din pis semnalul coresp din sig. Se va intoarce 0 in caz de succes si -1 caz de eroare.**
    6. **int myFunct(char**fin,char*fout,int n, int k) unde fin este un vector de n nume de fisiere. In fisierul cu numele dat de fout se va scrie al k-lea caracter din fiecare fisier cu numele dat de intrarile din fin. Se va intoarce 0 in caz de succes, -1 eroare.**

- **Grupa 242:**
    1. **Enumerati System IPC V.**
    2. **Enumerati 4 tipuri de fisiere.**
    3. **Spuneti pasii necesari accesarii unui disc.**
    4. **2 gestionare de fisiere sub windows.**
    5. **int myFunct(char *buf,int fd, int *poz), unde df este ID-ul unui fisier din descriptorul de fisiere al programului. Se citeste din fisier caracter si se pune in buf pe pozitia *poz, dupa care se incrementeaza circular *poz; returneaza 0 respectiv -1 daca a terminat cu success respectiv a dat de o eroare.**

```
#include <unistd.h>
// Includ biblioteca ce contine majoritatea apelurilor de sistem.
int myFunct(char *buf, int fd, int *poz){
        char temp;
// Citesc '1' caracter din fisierul mentionat de  descriptorul 'fd' si il
//pun in 'temp'.
    if( read(fd, &temp, 1) >= 0 )     {
        buf[*poz] = temp;
        (*poz)++;
        return 0;     }
 return -1;}
```

    7. **int myFunct(char *f1, char *f2, int m, int n) Se iau caracterele din fisierul f1 dintre pozitiile m si n si se pun la sfarsitul lui f2 (daca nu exista se creaza), se pun numai**

litere si cifre, restul caracterelor se sar; returneaza -1 in caz de eroare sau numarul de caractere puse in f2 in caz de succes.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
//fcntl.h contine constantele O_RDWR, O_RDONLY, O_CREAT, etc. si functia
//open()
int myFunct(char *f1, char *f2, int m, int n){
    int fd_1, fd_2;
//Cum in cerinta scrie ca trebuie sa iau elementele dintre 'm' si 'n',
//atunci trebuie sa fiu sigur ca 'm' este strict mai mic ca 'n'.Altfel,
//functia nu ar avea sens.
     if( m > n )
       return -1;
//Deschid fisierul 'f1' doar in citire si memorez descriptorul sau in fd_1
    if( (fd_1 = open(f1, O_RDONLY)) < 0)     {
         return -1;     }
//Deschid fisierul 'f2' in citire si scriere, avand niste optiuni
//suplimentare:
//O_RDWR = deschid fisierul in citire si scriere
//O_CREAT = creaza fisierul daca nu a fost creat
//O_APPEND = muta cursorul de citire/scriere la sfarsitul fisierului
//0644 = masca de drepturi pentru fisierul nou creat(daca nu a fost creat
//deja)
  if( (fd_2 = open(f2, O_RDWR | O_CREAT | O_APPEND, 0644)) < 0)     {
      return -1; }
//Mut cursorul de citire la pozitia 'm' a fisierului
     if( lseek(fd_1, m, SEEK_SET) < 0)     {
       return -1;     }
    int nr_de_char_scrisi = 0;
    while(m <= n)    {
 //Declar un char in care citesc din fisier.
    char temp;//Citesc un char.
  if( read(fd_1, &temp, 1) < 0 )     {
    return -1; }
//Verific daca char-ul citit este alfanumeric.Daca este, atunci il scriu
//in 'f2'.
        if(isalnum(temp))         {
              if( write(fd_2, &temp, 1) < 0 )    {
                  return -1;     }
              nr_de_char_scrisi++; }
        m++; }
       close(fd_1);
```

```
        close(fd_2);
        return nr_de_char_scrisi;}
```

Grupa 243:

1. sa se specifice apelul functiei care creaza un segment de memorie partajata care sa foloseasca IPC_ PRIVATE
2. sa se explice deadlock pentru 2 procese
3. sa se explice fenomenul de deadlock circular intre Pn procese
4. sa se specifice o strategie de evitare a deadlock- ului
5. sa se scrie o functie int myFunc(key_ t cheie) care sa transforme literele mici in litere mari si invers, din stringul de la inceputul segmentului de memorie desemnat de "cheie"

```
#include <ctype.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

int myFunct(key_t cheie)
{
    //Presupunem dimensiunea memoriei partajate ca fiind de 100,
    //Ã®ntrucÃ¢t Ã®n cerinÈ□Ä□ nu se zice cÃ¢t este.
    int dim = 100;

    //AccesÄ□m prin funcÈ□ia corespunzÄ□toare cu ajutorul cheii.
    int get;
    get = shmget(cheie, dim, 0666);
    if(get < 0)
    {
        return -1;
    }

    //AtaÈ□Ä□m segmentul de memorie partajat la variabila 'str'.
    char *str;
    str = (char *) shmat(get, NULL, 0);
    if(str == (void *) -1)
    {
        return -1;
    }

    //OdatÄ□ ataÈ□at, peste acest segment se pot desfÄ□È□ura operaÈ□ii
    //ca Ã®n orice alt vector alocat cu 'malloc'
```

```
        char *ss;
        for(ss = str; (*ss) != '\0'; ss++)
        {
            if(islower(*ss))  //caz literÄ micÄ
                (*ss) = toupper(*ss);
            else
                if(isupper(*ss)) //caz literÄ mare
                    (*ss) = tolower(*ss);
        }

        //DetaÈÄm segmentul de memorie.
        if(shmdt(str) < 0)
        {
            return -1;
        }

        return 0;
    }
```

1. sa se scrie o functie myFunc(char *f, char *s) care returneaza - 1 in caz de eroare, 1 daca primele caractere din fisierul f coincid cu cele din stringul s, si 0 altfel (asadar la problema asta trebuia sa verifici daca sirul de caractere s se gaseste la inceputul continutului fisierului desemnat de "f")

- note: un 10, vreo cinci de 9, cativa picati... iar majoritatea sub 8, avand in vedere ca au intrat la oral.

In majoritatea cazurilor oralul a scazut nota, si este de preferat ocolirea lui daca materia nu este stiuta perfect, si nu aveti indemanare peste medie in C

2008-2009

- Grupa 233

1. Procedura care permite initiliazarea unui proces cu alta prioritate decat cea initiala. Raspuns : nice
2. Numele algoritmului de gestionare a proceselor in Unix. Raspuns : Round Robin
3. Descrieti cel mai simplu scenariu de deadlock .
4. Enumerati doua tipuri de periferice.
5. o functie C care primea ca parametru id-ul lui vector de semafoare de tip mutex , un pointer la char si un pointer la functie. trebuia sa punem semaforul pe 0, sa apelam functia pointata

de al 3-lea parametru avand ca parametru pointerul la char si sa punem semaforul pe 1. succes return 0 , esec return -1

6. o functie int myStack(int *a,int max,int *n,int *val,int oper) care trebuia sa simuleze o stiva , a fiind adresa primului element din stiva, max numarul maxim de elemente pe care le poate avea stiva, n numarul curent de elemente si daca oper era 0 trebuia sa punem in stiva valoarea de la adresa val si daca era 1 sa extragem din stiva la adresa val. succes return 0, esec return -1

- Grupa 242
  1. Numiti algoritmul de criptare/decriptare care fol si chei private si publice (RSA)
  2. Numiti cele 3 categorii principale de virusi
  3. Numiti doua caracteristici ale proceselor in UNIX
  4. Care sunt drepturile asupra unui fisier in UNIX (3 drepturi)
  5. scrieri o functie int *myFunct(int *n) care asteapta terminarea tuturor celor proceselor fiu, pune intr-un vector pid-urile acestora si returneaza adresa acestuia. de asemenea *n va retine numarul de procese fiu

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

int *myFunct(int *n)
{
    //Nu È□tiu cÃ¢È□i fii are procesul, aÈ□a cÄ□ o sÄ□ presupun cÄ□ are maxim 256.
    int *copii = (int *) malloc(sizeof(int) * 256);

    //Ã□n 'wait_stat' memorez valoarea returnatÄ□ de 'wait()'
    int wait_stat;

    //Aici numÄ□r cÃ¢È□i copii avea procesul Ã®n total.
    int nr_de_copii = 0;

    while(1)
    {
        //AÈ□tept un fiu sÄ□ termine.
        wait_stat = wait(NULL);

        //Verific dacÄ□ wait-ul s-a desfÄ□È□urat cu succes.
```

```
        if(wait_stat < 0)
        {
            //Ã□n cazul Ã®n care 'wait()' eÈ□ueazÄ□ È□i seteazÄ□ errno cu
            //valoarea ECHILD, Ã®nseamnÄ□ cÄ□ toate procesele fiu s-au
terminat.
            if(errno = ECHILD)
            {
                //Setez *n-ul È□i returnez vectorul de piduri conform
cerinÈ□ei.
                *n = nr_de_copii;
                return copii;
            }

            //Returnez NULL dacÄ□ wait eÈ□ueazÄ□ cu oricare errno Ã®n
            //afarÄ□ de ECHILD.
            return NULL;
        }

        //Adaug pid-ul copilului "wait-at" Ã®n vector.
        copii[nr_de_copii] = wait_stat;
        nr_de_copii++;
    }
}
```

6. Scrieti o functie void myHand(int sig) care va instala handlerul de N ori(N definit cu #define) si la sfarsit instaleaza handlerul implicit.

2007-2008

Grupa 232

1.
2. Scrieti prototipul unei functii care scrie intr-un fisier sau la iesirea std.
3. Ce este memoria cache? (Nu a procesorului!)
4.
5. Scrieti o functie c: int funct(char* f1, char* f2, int m, int n) f1, f2: nume de fisiere, care copiaza de la pozitia m pana la pozitia n din fisierul f1 in fisierul f2 (pe care mi se pare ca il si creeaza). Returneaza 0 in caz de succes, -1 in caz de eroare. Atentie la parametri incorecti.->rez mai sus

2006-2007

- Grupa 244

1. Numiti 2 sisteme de operare.
2. Numiti 2 strategii de inlocuire a paginilor(gestiunea memoriei)
3. Explicati, pe scurt, notiunea de memorie virtuala.
4. Numiti 2 periferice la care accesul se face in mod bloc.
5. Scrieti o functie C: char *myfunct(int *p, char c), unde p este vector de 2 descriptori deschisi, asociati unui tub p[0] in citire si p[1] in scriere, care transmite pe tub caracterul c si citeste din tub un mesaj alcatuit astfel: primul caracter este lungimea mesajului urmand informatia (aici nu stiu exact oricum era ceva de genul: mesajul abc->3abc). Mesajul se depune in memorie si se returneaza adresa la care a fost depus. In caz de insucces, returneaza NULL.
6. Scrieti o functie C: int myfunct(int *pid, int *sig, int n), unde vectorii pid si sig au acelasi numar de elemente, si anume n. Se vatransmite fiecarui proces indentificat de elemntele lui pid semnalul corespunzator indentificat de elementul din sig. Functia returneaza 0 in caz de succes si valoarea lui errno in caz de eroare.

- Grupa 241
    1. Descrieti modalitatile de comunicare intre procese din IPC System V. Raspuns: cozi de mesaje, memorii partajate si semafoare
    2. Care este algorimtul de scheduling al proceselor? Raspuns: Round Robin
    3. Explicati concepul de deadlock.
    4. O pagina are 64 kb. Memoria are 128 MB. Cate pagini sunt in memorie?
    5. Scrieti o functie c: int funct(char* f1, char* f2, char *f3, char *fout) - parametrii sunt nume de fisiere - care face o interclasare intre f1,f2,f3 in fout astfel caracter din f1 caracter din f2 caracter din f3 caracter din f1 caracter din f2 caracter din f3 etc returneaza 0 in caz de succes, -1 in caz de eroare
    6. Scrieti o functie c int funct(char* f, char *word), f nume de fisier care cauta in f cuvantul de lungime minima si il pune in word; returneaza -1 in caz de eroare si pozitia in fisier a cuvantului de lungime minim altfel.