30k, 35k, 15k, 25k, 75k, 45k

39k, 40k, 33k, 20k, 21k

Foglalási igény	Memóriaterület-szabad terület					
	30	35	15	25	75	45
39	30	35	15	25	75	45
40	30	35	15	25	75	45
33	30	35	15	25	75	45
20	30	35	15	25	75	45
21	30	35	15	25	75	45

WORST FIT

Foglalási igény	Memóriaterület-szabad terület					
	30	35	15	25	75	45
39	30	35	15	25	39|36	45
40	30	35	15	25	75	40|5
33	30	35	15	25	33|3	40|5
20	30	20|15	15	25	33|3	40|5
21	21|9	35	15	25	33|3	40|5

Lefoglalja a legnagyobbat,

és mindig a legnagyobbat foglalja le.

FIRST FIT

Foglalási igény	Memóriaterület-szabad terület					
	30	35	15	25	75	45
39	30	35	15	25	36|39	45
40	30	35	15	25	75	5|40
33	30	2|33	15	25	75	45
20	10|20	35	15	25	75	45
21	30	35	15	4|21	75	45

Előről indul,
és a legelső szabad helyre megy.

NEXT FIT

Foglalási igény	Memóriaterület-szabad terület					
	30	35	15	25	75	45
39	30	35	15	25	39|36	45
40	30	35	15	25	75	40|5
33	30	33|2	15	25	75	45
20	30	35	15	20|5	75	45
21	30	35	15	25	21|15	45

Keres egy helyet, lefoglalja majd utána elindul
onnan tovább és ha elfogy a szabad terület
újrakezdi.

BEST FIT

Foglalási igény	Memóriaterület-szabad terület					
	30	35	15	25	75	45
39	30	35	15	25	75	39|6
40	30	35	15	25	40|35	45
33	30	33|2	15	25	75	45
20	30	35	15	20|5	75	45
21	21|9	35	15	25	75	45

Azokat a helyeket keresi ahol a legkevesebb,
Meghagyja a maradék szabad területet.

semset.c (~/Asztal/C)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semset.c × semval.c × semup.c × semkill.c ×

```c
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x12  // Kulcs

    int semid,  // Szemafor azonosító
        nsems,  // Szemaforok száma
        semnum, // Szemafor száma
        rtn;    // Visszatérési érték

    int semflg; // Flag
    struct sembuf sembuf, *sop;
    union semun;

    int cmd;    // semctl()-nek szóló parancs változója

int main()
{

    int arg;


    nsems = 1;  // Egyetlen szemafor a set-ben
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("\n Az azonosító: %d\n",semid);
    printf ("Kérem a semval erteket: ");

    semnum = 0; // 0. szemafort azonosítom

    cmd = SETVAL;   // Állítsd be a szemafor érteket
    scanf("%d", &arg);
    rtn = semctl(semid, semnum, cmd, arg); // Semid-vel azonosított set 0-ik szemafor

    printf("\nVisszatérési érték: %d\nSemval értéke: %d\n",rtn, arg);

    return 0;

}
```

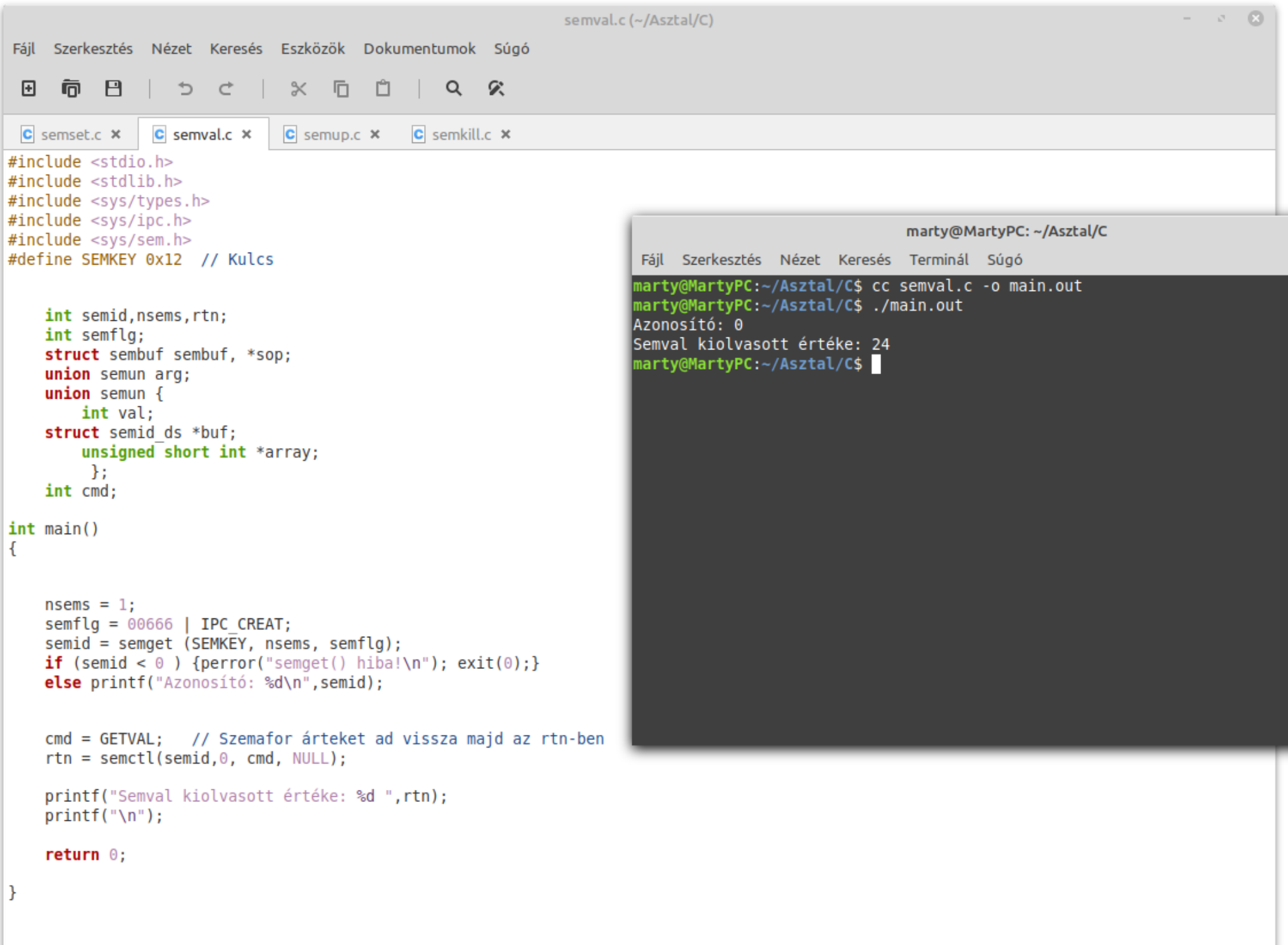
marty@MartyPC: ~/Asztal/C

Fájl Szerkesztés Nézet Keresés Terminál Súgó

```
marty@MartyPC:~/Asztal/C$ cc  semset.c -o main.out
marty@MartyPC:~/Asztal/C$ ./main.out

 Az azonosító: 0
Kérem a semval erteket: 24

Visszatérési érték: 0
Semval értéke: 24
marty@MartyPC:~/Asztal/C$
```


semval.c (~/Asztal/C)
Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó
semset.c
semval.c
semup.c
semkill.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x12 // Kulcs
int semid,nsems,rtn;
int semflg;
struct sembuf sembuf, *sop;
union semun arg;
union semun {
int val;
struct semid_ds *buf;
unsigned short int *array;
};
int cmd;
int main()
{
nsems = 1;
semflg = 00666 | IPC_CREAT;
semid = semget (SEMKEY, nsems, semflg);
if (semid < 0) {perror("semget() hiba!\n"); exit(0);}
else printf("Azonosító: %d\n",semid);
cmd = GETVAL; // Szemafor árteket ad vissza majd az rtn-ben
rtn = semctl(semid,0, cmd, NULL);
printf("Semval kiolvasott értéke: %d ",rtn);
printf("\n");
return 0;
}
marty@MartyPC: ~/Asztal/C
Fájl Szerkesztés Nézet Keresés Terminál Súgó
marty@MartyPC:~/Asztal/C$ cc semval.c -o main.out
marty@MartyPC:~/Asztal/C$ ./main.out
Azonosító: 0
Semval kiolvasott értéke: 24
marty@MartyPC:~/Asztal/C$

semup.c (~/Asztal/C)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semset.c × semval.c × semup.c × semkill.c ×

```c
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x12  // Kulcs

    int semid, nsems, rtn;
    unsigned nsops; // Ezzel adjuk meg, hány struktúrával foglakozzon
    int semflg;
    struct sembuf sembuf, *sop;


int main()
{


    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("Azonosító: %d\n",semid);
    printf ("\n");

    nsops = 1;      //Egy művelet van
    sembuf.sem_num = 0; //A 0-ik szemaforral foglakozunk
    sembuf.sem_op = 1;  // Növelés
    sembuf.sem_flg = 0666;  // Flag beállítás
    sop = &sembuf;      // Argumentum kérése
    rtn = semop(semid, sop, nsops); // A 0-val visszatérő semop sikeres.
    printf("Visszatérési érték: %d\n",rtn);

    return 0;

}
```

marty@MartyPC: ~/Asztal/C

Fájl Szerkesztés Nézet Keresés Terminál Súgó

```
marty@MartyPC:~/Asztal/C$ cc semup.c -o main.out
marty@MartyPC:~/Asztal/C$ ./main.out
Azonosító: 0

Visszatérési érték: 0
marty@MartyPC:~/Asztal/C$
```

semkill.c (~/Asztal/C)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

semset.c × semval.c × semup.c × semkill.c ×

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#define SEMKEY 0x10  // Kulcs


    int semid, nsems, rtn;
    int semflg;
    struct sembuf sembuf, *sop;
    union semun {
        int val;
    struct semid_ds *buf;
        unsigned short int *array;
         };
    int cmd;

int main()
{

    int arg;
    nsems = 1;
    semflg = 00666 | IPC_CREAT;
    semid = semget (SEMKEY, nsems, semflg);
    if (semid < 0 ) {perror("semget() hiba!\n"); exit(0);}
    else printf("semid értéke: %d\n",semid);

    cmd = IPC_RMID;     // Ezzel szüntetjük meg
    rtn = semctl(semid, 0, cmd, arg);

    printf("Kill visszatérés: %d\n",rtn);

    return 0;

}
```

marty@MartyPC: ~/Asztal/C

Fájl Szerkesztés Nézet Keresés Terminál Súgó

```
marty@MartyPC:~/Asztal/C$ cc semkill.c -o main.out
marty@MartyPC:~/Asztal/C$ ./main.out
semid értéke: 1
Kill visszatérés: 0
marty@MartyPC:~/Asztal/C$
```

gyak11_2.c (~/Asztal/C)

Fájl Szerkesztés Nézet Keresés Eszközök Dokumentumok Súgó

gyak11_2.c ×

```c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

sem_t m;

void* thread(void* arg)
{
    //wait
    sem_wait(&m);
    printf("A pidem: %d\nVárakozás...\n", getpid());  // PID kiíratás
    sleep(3);   // Várakozás
    sem_post(&m);
}


int main()
{
    sem_init(&m, 0, 1);
    pthread_t t1,t2,t3;  // 3 feladat
    pthread_create(&t1, NULL,thread,NULL);
    pthread_create(&t2, NULL,thread,NULL);
    pthread_create(&t3, NULL,thread,NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);
    sem_destroy(&m);    // Megszüntetés
    return 0;
}
```

marty@MartyPC: ~/Asztal/C

Fájl Szerkesztés Nézet Keresés Terminál Súgó

```
marty@MartyPC:~/Asztal/C$ cc -pthread gyak11_2.c -o main.out
marty@MartyPC:~/Asztal/C$ ./main.out
A pidem: 2676
Várakozás...
A pidem: 2676
Várakozás...
A pidem: 2676
Várakozás...
marty@MartyPC:~/Asztal/C$
```