

Magiszter

Számítástechnikai szerkesztőség

CEX MS és Aztec C

alatti
Programozási könyvtár

Programvédelem nélkül!

Szoftver
IBM, PC, XT, AT
kompatibilis mikroszámítógépekre



BUDAPEST
1828



AKADÉMIA KIADÓ

SCHILL R. SCHILL—SZEGEDI EDI JÁNOS

CEX

Kiterjesztett C könyvtár

Kiterjesztett C könyvtár

Microsoft C (+4.00 változat) és

Artex Mark II (+3.20 változat)

nyelvrakás

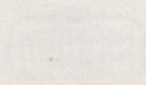
3.2 változat

MS-DOS 2.XX

MS-DOS 3.XX

12/8

NTM



Akadémiai Kiadó, Budapest, 1982

SCHILL RÓBERT—SZEGEDI JÁNOS

CEX

Kiterjesztett C könyvtár

Microsoft C (4.00 változat) és

Aztec Manx C (3.20 változat)

nyelvekhez

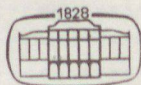
2.2 Változat

MS—DOS 2.XX

MS—DOS 3.XX

1310

MTAK



Akadémiai Kiadó, Budapest 1987

660922

Szoftver-dokumentáció

Sorozatszerkesztő: Pintér Tibor

A kiadványt szakmailag ellenőrizte: Meggyesházi János

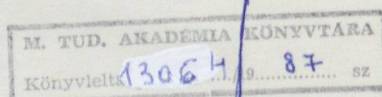
©Akadémiai Kiadó és Nyomda · Schill Róbert—Szegedi János, 1987

Minden jog fenntartva

MAGYAR
AKADÉMIAI KÖNYVTÁR

A szoftver-dokumentáció a DOG dokumentáció generátorral készült

©Akadémiai Kiadó és Nyomda · Schill—Szegedi, 1987



T a r t a l o m

1. Bevezetés.....	8
2. A CEX könyvtár felépítése.....	10
2.1. A képernyő kezelése.....	10
2.2. A képernyő ablakos kezelése.....	11
2.3. Általánosított teljesképernyős adatkezelés.....	11
2.4. A billentyűzet kezelése.....	11
2.5. Lemez, alkönyvtár és adatállomány kezelése.....	12
2.6. A hangszóró kezelése.....	12
2.7. RS232 kommunikáció.....	13
2.8. Bittábla manipuláció.....	13
2.9. C nyelv - operációs rendszer könnyítés.....	13
2.10. Dátum, idő.....	13
2.11. Aztec C-ből hiányzó string-függvények.....	13
3. A CEX könyvtári függvények leírása.....	14
3.1. AERASE.....	14
3.2. AFRAME.....	15
3.3. AGETTYS.....	16
3.4. APUTTY.....	17
3.5. APUTTYS.....	18
3.6. APUTTYSN.....	19
3.7. ASKBIT.....	20
3.8. AWRCHAR.....	21
3.9. BEEP.....	22
3.10. CLEARBIT.....	23
3.11. CLEARBTAB.....	23
3.12. COLDOUBLE.....	24
3.13. COLSIMPLE.....	25
3.14. CREABTAB.....	26
3.15. CUPO.....	27
3.16. CURRDISK.....	28
3.17. CURSTYPE.....	28
3.18. DELAY.....	29
3.19. EDITLINE.....	30
3.20. ENDMUSIC.....	31
3.21. EOL.....	31
3.22. EOP.....	33
3.23. ERASE.....	33
3.24. FCOPY.....	34
3.25. FRAME.....	35

3.26. FREEBTAB.....	36
3.27. FREEDISK.....	37
3.28. FRSTFILE.....	37
3.29. GETCPOS.....	40
3.30. GETDAT.....	40
3.31. GETDIR.....	41
3.32. GETMODE.....	42
3.33. GETPAGE.....	43
3.34. GETTIM.....	44
3.35. GETTY.....	44
3.36. GETTYS.....	46
3.37. GETVOL.....	47
3.38. INIMUSIC.....	48
3.39. KBFLUSH.....	50
3.40. KBGET.....	50
3.41. KBSHIFTS.....	51
3.42. KBSTAT.....	52
3.43. LINDOUBLE.....	53
3.44. LINSIMPLE.....	54
3.45. MOVESCR.....	55
3.46. MOVFRAME.....	55
3.47. MUSIC.....	57
3.48. NEXTFILE.....	58
3.49. NOSOUND.....	59
3.50. PARSNAME.....	59
3.51. PARSPATH.....	61
3.52. PRINHELP.....	61
3.53. PROMPT.....	63
3.54. PUTTY.....	64
3.55. PUTTYS.....	65
3.56. RDCHAR.....	65
3.57. RS232INIT.....	66
3.58. RS232RECE.....	66
3.59. RS232SEND.....	67
3.60. RS232STAT.....	68
3.61. SELECT.....	68
3.62. SELFIE.....	71
3.63. SETATTR.....	73
3.64. SETBIT.....	75
3.65. SETBTAB.....	75
3.66. SETCPOS.....	76

3.67. SETMODE.....	77
3.68. SETPAGE.....	78
3.69. SOUND.....	79
3.70. STRDUP.....	79
3.71. STRLWR.....	80
3.72. STRUPR.....	81
3.73. WINDCLOSE.....	82
3.74. WINDFRAM.....	83
3.75. WINDFREE.....	84
3.76. WINDMOVE.....	84
3.77. WINDREST.....	85
3.78. WINDSAVE.....	86
3.79. WRCHAR.....	88
3.80. XSELECT.....	88
4. Eltérések az MSCEX és ACEX környezetek között.....	95
5. A demonstrációs programok forráslistája.....	96
5.1. DEMO.C.....	97
5.2. DFILEDIR.C.....	107
5.3. DBACKMUS.C.....	112
5.4. DKEYSCR.C.....	114

1. Bevezetés

A CEX könyvtár C nyelvből hívható hasznos rutinok gyűjteménye, amelyek a DOS operációs rendszer BDOS és BIOS rendszereihez adnak kényelmes programozói interfészt, valamint programozói gyakorlatban hasznos funkciók általánosított algoritmusát valósítják meg.

A CEX elsősorban olyan gyakorlott programozók munkáját segíti, akik rendszerprogramokat vagy fejlett alkalmazói programokat állítanak elő.

A CEX elősegíti az egységes felhasználói interfész megvalósítását.

A CEX disztributív lemezen a következő könyvtárak találhatók:

LIBS
HEADERS
DEMO

A könyvtárak tartalma:

- | | | |
|-----------|--|---|
| - LIBS | ACEX.LIB
MSCEX.LIB | az Aztec run-time könyvtár
a Microsoft run-time könyvtár |
| - HEADERS | AZTEC.H
BITTAB.H
EDITLINE.H
FCOPY.H
FRAME.H
FRSTFILE.H
KBGET.H
KBSHIFTS.H
MUSIC.H
PARSNAME.H
RS232.H
SELECT.H
SELFIE.H
SETATTR.H
WINDOW.H
XSELECT.H | |

- DEMO	DEMO.EXE	demonstrációs monitor
	DEMO.C	- " - forrása
	DBACKMUS.EXE	háttérzene
	DBACKMUS.C	- " - forrása
	DFILEDIR.EXE	file- és könyvtárkezelés
	DFILEDIR.C	- " - forrása
	DKEYSCR.EXE	billentyűzet- és képernyőkezelés
	DKEYSCR.C	- " - forrása

A CEX mindkét nyelvhez alkalmazási területének megfelelően csak SMALL memóriamodellben létezik.

A könyvtárak beépítése érthetőszerű: a felhasználó tárgy kódú moduljaihoz hozzá kell szerkeszteni (linkelni).

2. A CEX könyvtár felépítése

A CEX a rutinok következő csoportjait tartalmazza:

- Képernyőkezelés
- Ablakos képernyőkezelés
- Ablakos help kiírása
- Altalánosított teljesképernyős adatkezelés
- Billentyűzet kezelése
- Lemez, alkönyvtár, adatállomány kezelése
- Hangszóró kezelése
- RS232 kommunikáció megvalósítása
- Bittábla manipulációk
- C nyelv - operációs rendszer könnyítés
- Dátum, idő
- Aztec C-ből hiányzó string-függvények

2.1. A képernyő kezelése

törlések

erase, aerase, eop, eol

pozicionálás

cupo, setcpos, getcpos

kiírások

putty, aputty, aputtys, aputtysn, wrchar, awrchar, ...

keretezés

frame, aframe, lindouble, linsimple, coldouble, colsimple

mozgatás

movframe, movescr

olvasás képernyőről

rdchar

képernyőkezelés paraméterezése
getmode, setmode, getpage, setpage, curstype

2.2. A képernyő ablakos kezelése

ablak nyitása
windsave, windfram

ablak lezárása
windrest + windfree, windclose

ablak mozgatása
windmove

2.3. Altalánosított teljesképernyős adatkezelés

egysoros szövegszerkesztés
editline

minta alapján történő ablakos file-kiválasztás
selfile

ablakos adatkezelés
select, xselect

ablakos help kiírása
printhelp

2.4. A billentyűzet kezelése

billentyűzet olvasása
getty, kbgget

pufferelt beolvasás egyidejű kiírással
gettys, agettys

billentyűzet státusza
kbstat, kbflush

billentyűzet SHIFT állapota
kbshifts

2.5. Lemez, alkönyvtár és adatállomány kezelése

lemez-címke
getvol

aktuális lemezmeghajtó
currdisk

aktuális alkönyvtár
getdir

lemezen lévő szabad hely
freedisk

file-név, könyvtárnév elemzése
parsname, parspath

könyvtár tartalomjegyzékének kezelése
frstfile, nextfile

file másolása
fcopy

2.6. A hangszóró kezelése

hangszóró megszólaltatása
sound, delay, nosound

zene
music, inmusic, endmusic

hangjelzés
beep

2.7. RS232 kommunikáció

Inicializálás, fogadás, küldés, státusz
rs232init,rs232rece,rs232send,rs232stat

2.8. Bittábla manipuláció

létrehozás, beállítások, törlések, lekérdezések
creabtab,setbtab,clearbtab,askbit,setbit,clearbit,freebtab

2.9. C nyelv - operációs rendszer könnyítés

ARGC,ARGV programparaméterek beolvasása betöltött programból
prompt

2.10. Dátum, idő

dátum beszédése formátumozottan
getdat

idő beszédése formátumozottan
gettim

2.11. Aztec C-ből hiányzó string-függvények

string dinamikus kettőzése
strdup

betűkonverzió stringben
strlwr,strupr

3. A CEX könyvtári függvények leírása

A következő alfejezetekben az egyes könyvtári függvények (és a hozzájuk tartozó header file-ok) leírása következik. A C nyelv szabályainak megfelelően a header file-ok neve mindig a < és > jelek között szerepel. A header file-ok közül csak a nem triviális tartalmúakat ismertetjük a megfelelő nevű függvény leírásánál.

A leírások szerkezete a következő:

Összegzés

Leírás

Visszatérési érték

Lásd még (kapcsolódó függvények, header file-ok)

Példa

A "Példa" rész nem szerepel azoknál a függvényeknél, ahol a működés egyértelmű.

3.1. AERASE

Összefoglalás

```
#include <setattr.h>
```

```
aerasett(attr)
```

```
int attr;
```

a kívánt színezés

Leírás

Törli a képernyőt és a megadott színre állítja a karaktereket. Törlés után a kurzor pozíciója 0,0. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

eof(), eop(), erase(), setattr(), <setattr.h>

Példa

```
#include <setattr.h>
.
.
.
aerase(GRAY);          /* torles, fekete alapon szurke betu */
```

3.2. AFRAME

Összefoglalás

```
#include <frame.h>
#include <setattr.h>
```

```
aframe(ulr,ulc,lrr,lrc,ftype,attr)
    int ulr;          a keret felső sora
    int ulc;          a keret bal oszlopa
    int lrr;          a keret alsó sora
    int lrc;          a keret jobb oszlopa
    int ftype;        a keret típusa
    int attr;         a kívánt színezés
```

Leírás

A megadott képernyőrészre keretet rajzol alfanumerikus üzemmódban. A keretező karakterek színezését az 'attr' paraméter határozza meg. A lehetséges kerettípusok a következők (definíciójuk a <frame.h> header file-ban található):

HDVD (0)	dupla keret
HSVS (1)	szimpla keret
HSVD (2)	vízszintesen szimpla, függőlegesen dupla
HDVS (3)	vízszintesen dupla, függőlegesen szimpla
HNVN (4)	üres keret

A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

frame(), movframe(), <frame.h>, <setattr.h>

Példa

```
#include <frame.h>
#include <setattr.h>
aframe(0,0,24,79,HDVD,BLUE);    /* egy teljes kek dupla keret */
```

3.3. AGETTYS

Összefoglalás

```
#include <setattr.h>
```

```
agettys(buffer,max,attr)
```

```
    char *buffer;
```

```
    int max;
```

```
    int attr;
```

az output puffer

a beolvasható karakterek száma

a kívánt színezés

Leírás

A billentyűzetről a pufferbe olvassa (és a megadott színnel kiírja) a karaktereket mindaddig, amíg ENTER-t vagy ESC-et nem kap. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

ESC (33 oktális, 27 decimális, 1B hexa), ha ESC zárta (ekkor az output puffer üres), egyébként 0 (ekkor az output pufferben a 0-val lezárt string található).

Lásd még

agetty(), getty(), gettys(), <setattr.h>

Példa

```
#include <setattr.h>
#define MAXNAME 50

char name[MAXNAME+1];

cupo(0,0);
aputtys("Adja meg a file nevet: ",GRAY);
if (agetts(name,MAXNAME,WHITE) == '\33')
    return(0);

.
.
.
```

3.4. APUTTY

Összefoglalás

```
#include <setattr.h>
```

```
aputty(c,attr)
```

```
    char c;
```

a kiírandó karakter

```
    int attr;
```

a kívánt színezés

Leírás

Kiírja a karaktert a képernyőre a megadott színnel. A kurzor egy pozícióval jobbra mozdul. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aputtys(), awrchar(), putty(), puttys(), wrchar(), <setattr.h>

Példa

```
#include <setattr.h>
int color,c;

c = 'A';          /* betuk kiirasa kulonbozo szinekkal    */
for (color = LGRAY; color <= WHITE; color++)
    aputtys(c++,color);
```

3.5. APUTTYS

Összefoglalás

```
#include <setattr.h>
```

```
aputtys(str, attr)
```

```
    char *str;
    int attr;
```

a kiírandó string
a kívánt színezés

Leírás

Kiírja a stringet a képernyőre a megadott színnel. Kiírás után a kurzor a kiírt string mögött áll. A CR - LF sorvégjelek a 80-as pozíción figyelmen kívül maradnak. A rutin a karaktereket közvetlenül a képernyőpufferba írja, ezért ez a kiírási mód nagyon gyors. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aputty(), aputtysn(), awrchar(), putty(), puttys(), wrchar(), <setattr.h>

Példa

```
#include <setattr.h>

char *szinek[] = {
    "kek",
    "zold",
    "vilagoskek",
    "piros",
    "lila",
    "barna",
    "vilagosszurke",
    0};

char **szp;
int lin,col,color;

lin = 0;
col = 10;
color = BLUE;
erase();
for (szp = szinek; *szp; szp++)
{
    cupo(lin++,col);
    aputtys(*szp,color++);
}
```

3.6. APUTTYSN

Összefoglalás

```
#include <setattr.h>
```

```
aputtysn(str, attr, len)
```

```
char *str;
```

```
int attr;
```

```
int len;
```

a kiírandó string

a kívánt színezés

a kitöltendő hossz

Leírás

Kiírja a stringet a képernyőre a megadott színezéssel, a megadott hosszon. A kurzor helye nem változik. Ha a string hossza nagyobb a megadott hosszánál, akkor csak a string eleje kerül kiírásra. Ha a string rövidebb, mint a megadott hossz, akkor a maradék helyek szóközökkel töltődnek fel. A kiírandó stringnek az aktuális sorban el kell férnie. A rutin a karaktereket közvetlenül a képernyőpufferba írja, ezért ez a kiírási mód nagyon gyors. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aputty(), aputtys(), awrchar(), putty(), puttys(), wrchar(),
<setattr.h>

Példa

```
#include <setattr.h>
```

```
char hosszu[] = "ez egy hosszu szoveg";  
char rovid[] = "rovid";
```

```
erase();  
cupo(0,0);      aputtysn(hosszu,RED,10);  
cupo(1,0);      aputtysn(rovid,RED,10);
```

3.7. ASKBIT

Összefoglalás

```
#include <bittab.h>
```

```
int askbit(lin,col,pbt,size)
```



```
int lin,col;
char *pbt;
int size;
```

a bit koordinátái
a bittábla mutatója
a bittábla mérete

Leírás

A megadott bittábla megadott bitjének értékét kérdezi le.

Visszatérési érték

A megadott bit értéke.

Lásd még

clearbit(), clearbtabs(), creabtab(), freebtabs(), setbit(),
setbtabs()

Példa

Lásd a 3.14 CREABTAB c. pontnál.

3.8. AWRCHAR

Összefoglalás

```
#include <setattr.h>
```

```
awrchar(num,ch,attr)
```

```
    unsigned num;
```

```
    char ch;
```

```
    int attr;
```

a karakter ismétlési száma

a kiírandó karakter

a kívánt színezés

Leírás

Kiírja a karaktert a képernyőre a megadott színnel az aktuális pozíciótól kezdve annyiszor, amennyi az ismétlési tényező. A kurzor pozíciója nem változik. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

wrchar(), <setattr.h>

Példa

```
#include <setattr.h>
```

```
cupo(23,0); /* utolso ket sor */
awrchar(160,' ',BLUE); /* torlese kek szinure */
```

3.9. BEEP

Összefoglalás

```
beep(start,end,time)
    int start;          a kezdő hangmagasság (Hz)
    int end;            a végső hangmagasság (Hz)
    int time;           az időtartam (1/100 s)
```

Leírás

A megadott időn keresztül megszólaltatja a hangszórót a kezdőtől a végső hangmagasságig.

Visszatérési érték

Nincs.

Lásd még

backmusic(), music(), sound()

Példa

```
beep(50,200,100); /* hangjelzes 50-200 Hz-ig 1 s-on */
```


3.10. CLEARBIT

Összefoglalás

```
#include <bittab.h>
```

```
clearbit(lin,col,pbt,size)
```

```
int lin,col;
```

```
char *pbt;
```

```
int size;
```

a bit koordinátái

a bittábla mutatója

a bittábla mérete

Leírás

Hamisra állítja a megadott bitet.

Visszatérési érték

Nincs.

Lásd még

askbit(), clearbtabs(), creabtab(), freebtabs(), setbit(),
setbtabs()

Példa

Lásd a 3.14 CREABTAB c. pontnál.

3.11. CLEARBTAB

Összefoglalás

```
#include <bittab.h>
```

```
clearbtabs(pbt,size)
```

```
char *pbt;
```

```
int size;
```

a bittábla mutatója

a bittábla mérete

Leírás

A CREABTAB függvénnyel létrehozott bittábla összes bitjét hamisra állítja.

Visszatérési érték

Nincs.

Lásd még

askbit(), clearbit(), creabtab(), freebtab(), setbit(),
setbtab()

Példa

Lásd a 3.14 CREABTAB c. pontnál.

3.12. COLDDOUBLE

Összefoglalás

```
#include <setattr.h>
```

```
coldouble(lin,col,len,attr)
```

```
int lin;
```

a kezdő sor a képernyőn

```
int col;
```

az oszlop

```
int len;
```

a vonal hossza

```
int attr;
```

a kívánt szín

Leírás

A képernyőn lévő keretelemek közé beillesztve egy dupla függőleges vonalat rajzol alfanumerikus üzemmódban. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aframe(), colsimple(), frame(), lindouble(), linsimple(),
<frame.h>, <setattr.h>

Példa

```
#include <setattr.h>
#include <frame.h>

aframe(0,10,20,40,HVD,GRAY); /* keretezes */
coldouble(0,25,21,GRAY); /* a keret ketteosztasa */
```

3.13. COLSIMPLE

Összefoglalás

```
#include <setattr.h>
```

```
colsimple(lin,col,len,attr)
    int lin;          a kezdő sor
    int col;          az oszlop
    int len;          a vonal hossza
    int attr;         a kívánt szín
```

Leírás

A képernyőn lévő keretelemek közé beillesztve egy függőleges vonalat rajzol alfanumerikus üzemmódban. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aframe(), coldouble(), frame(), lindouble(), linsimple(),
<frame.h>, <setattr.h>

Példa

```
#include <setattr.h>
#include <frame.h>

aframe(0,10,20,40,HDVS,GRAY); /* keretezes */
colsimple(0,25,21,GRAY); /* a keret ketteosztasa */
```

3.14. CREABTAB

Összefoglalás

```
#include <bittab.h>

char *creabtab(size)
    int size;                a létrehozandó bittábla mérete
```

Leírás

Dinamikus memóriaterületet foglal le a bittábla számára.

Visszatérési érték

Ha nincs rendelkezésre álló memória akkor 0, egyébként a bittábla mutatója.

Lásd még

askbit(), clearbit() clearbtabs(), freebtabs(), setbit(),
setbtabs()

Példa

```
#include <bittab.h>

char *bt;
int i,j;

if ((bt = creabtab(80)) == 0) /* 80 x 80-as bittabla */
    return(-1);
```



```

clearbtabs(bt,80);                                /* hamisra allitjuk */
process(bt,....);                                  /* feldolgozas */

/* tablázat nyomtatás */

for (i = 0; i < 80; i++)
{
    for (j = 0; j < 80; j++)
    {
        if (askbit(i,j,bt,80))
            aputty('*',WHITE);
        else
            aputty(' ',GRAY);
    }
    putty("\r\n");                                /* következó sor */
}

```

3.15. CUP0

Összefoglalás

```

cupo(row,col)
    int row;          a kívánt sor
    int col;          a kívánt oszlop

```

Leírás

A kurzort a megadott pozícióra állítja. A képernyő bal felső sarka jelenti a 0,0 pozíciót.

Visszatérési érték

Nincs.

Lásd még

getcpo(), setcpo()

Példa

```

cupo(0,0);          /* bal felsó sarok */
cupo(24,79);        /* jobb alsó sarok 80x25-os modban */

```


3.16. CURDISK

Összefoglalás

currdisk()

Leírás

Visszaadja az aktuális lemezmeghajtó sorszámát.

Visszatérési érték

Az aktuális lemezmeghajtó sorszáma. (A: = 0, B: = 1...)

Példa

```
int c;  
  
c = currdisk();  
printf("Az aktualis meghajto a %c:", c+'A');
```

3.17. CURSTYPE

Összefoglalás

```
unsigned curstype(start,end)  
    int start;           a kurzor felső pontsora  
    int end;             a kurzor alsó pontsora
```

Leírás

A kurzor típusát a megadott értékre állítja, majd visszaadja a korábbi kurzortípust.

Visszatérési érték

A korábbi kurzortípus (felső-alsó byte).

Lásd még

```
aerase(), awrchar(), eol(), erase()
```

Példa

```

.
.
unsigned savecurs, curstype();

savecurs = curstype(-1,-1);          /* kurzor eltuntetese */
.
.
.
curstype(savecurs >> 8, savecurs & 0xFF); /* visszaallitas */

```

3.18. DELAY

Összefoglalás

```
delay(count)
    int count;                a várakozás ideje (1/1000 s)
```

Leirás

A program futását felfüggeszti a megadott ezredmásodpercig (normál XT-re beállítva).

Visszatérési érték

Nincs.

Lásd még

nosound(), sound()

Példa

```
sound(500);      /* a hangszoro bekapcsolasa 500 Hz hangon      */
```



```
delay(3000);      /* 3 másodpercig hallható          */
nosound();        /* a hangszóró kikapcsolása          */
```

3.19. EDITLINE

Összefoglalás

```
#include <editline.h>
#include <setattr.h>
```

```
char *editline(source, lastcol, max, editc, helpc, prech)
    char *source;          a szerkesztendő szöveg
    int lastcol;           a szerkesztési sor utolsó oszlopa
    int max;               az output szöveg maximális hossza
    int editc;             a szerkesztési szín
    int helpc;             a help szín
    int prech;             az előre leütött karakter kódja
```

Leírás

Az editline egy egysoros szövegszerkesztő. A rutin segítségével maximálisan 255 karakter hosszúságú stringek olvashatók be illetve módosíthatók. A szerkesztési mező az aktuális pozíciótól a megadott utolsó oszlopig tart, de az ennél hosszabb szövegek is módosíthatók, mivel a szerkesztési mező csak egy ablak a szövegre, amelyben a szöveg a szükséges irányba elmozdulhat. A szerkesztés után a kurzor pozíciója nem változik. Ha az előre leütött karakter nem Carriage Return (13-as karakter), akkor a rutin ezt veszi be első leütött karakterként. Az input szöveg a szerkesztés során nem változik. A módosított szöveg egy statikus pufferbe kerül, amely a következő EDITLINE hívásig megőrzi tartalmát. A lehetséges színnevezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Az EDITLINE funkciói:

```
F1:      segítő információ angolul
ENTER:   kiszállás, a szerkesztés befejezése
ESC:     kiszállás a változtatás törlésével
```


HOME: ugrás a szöveg első karakterére
 END: ugrás a szöveg utolsó karakterére
 DEL: az aktuális karakter törlése
 CTR_D: az összes karakter törlése
 CTR_DEL: - " -
 BS: az előző karakter törlése
 INS: beszúrás/felülírás váltása

Visszatérési érték

EL_ESC - ESC leütése esetén,
 EL_NUL - ha az output string üres,
 egyébként az output stringet tartalmazó statikus puffer
 mutatója.

Lásd még

agetty(), getty(), <editline.h>, <setattr.h>

Példa

```
#include <editline.h>
#include <setattr.h>

char *p;

cupo(24,0);
puttys("Add meg a file nevet ");
p = editline("*.DAT",79,80,GRAY,WHITE,'\r');
if (p == EL_ESC)
    return;
if (p == EL_NUL)
    p = " *.*";
process(p);
```

3.20. ENDMUSIC

Összefoglalás

```
#include <music.h>
```


endmusic()

Leírás

Az INIMUSIC függvény által elindított háttérzenét állítja le.

Visszatérési érték

Nincs.

Lásd még

inimusic(), music(), <music.h>

Példa

Lásd az INIMUSIC függvény leírásánál.

3.21. EOL

Összefoglalás

eol()

Leírás

Az aktuális pozíciótól törli a sort. A kurzor pozíciója változatlan marad.

Visszatérési érték

Nincs.

Lásd még

aerase(), awrchar(), eop(), erase()

3.22. EOP

Összefoglalás

eop()

Leírás

Az aktuális pozíciótól törli a sort és az alatta lévő sorokat.
A kurzor pozíciója változatlan marad.

Visszatérési érték

Nincs.

Lásd még

aerase(), awrchar(), eol(), erase()

3.23. ERASE

Összefoglalás

erase()

Leírás

Törli a képernyőt, majd a kurzort a képernyő bal felső sarkába állítja.

Visszatérési érték

Nincs.

Lásd még

aerase(), eol(), eop()

3.24. FCOPY

Összefoglalás

```
#include <fcopy.h>
```

```
int fcopy(src,dst,buffer,bsize)
```

char *src;	a forrás-file neve
char *dst;	a cél-file neve
char *buffer;	a másolási puffer mutatója
unsigned bsize;	a másolási puffer mérete

Leírás

Olyan (gyors) belső file-másoló, amely a file dátumát és attribútumait is átviszi. A másolási puffer (és annak mérete) a felhasználó lehetőségeitől függ.

Visszatérési érték

- 0 - ha a másolás sikeres,
- 1 - ha a forrás-file nem nyitható meg,
- 2 - ha a cél-file nem nyitható meg,
- 3 - ha olvasási hiba fordult elő,
- 4 - ha írási hiba fordult elő.

A hibakódok leírása az <fcopy.h> header file-ban található.

Példa

```
#include <fcopy.h>
```

```
#define MAXBUF 32767
```

```
char buffer[MAXBUF];
```

```
char *fcopyerr[] =
```

```
{  
    "Sikeres masolas",  
    "Nyitási hiba a forras-file-nal",
```



```

    "Nyitási hiba a cel-file-nal",
    "Olvasási hiba",
    "Írasi hiba"
};

```

```

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

    printf("%s masolasa a %s file-ba: ",argv[1],argv[2]);
    res = fcopy(argv[1],argv[2],buffer,MAXBUF);
    printf(fcopyerr[res]);
}

```

3.25. FRAME

Összefoglalás

```
#include <frame.h>
```

```

frame(ulr,ulc,lrr,lrc,ftype)
    int ulr;           a keret felső sora
    int ulc;           a keret bal oszlopa
    int lrr;           a keret alsó sora
    int lrc;           a keret jobb oszlopa
    int ftype;         a keret típusa

```

Leírás

A megadott képernyőrészre keretet rajzol alfanumerikus üzemmódban. A lehetséges kerettípusok a következők (definíciójuk a <frame.h> header file-ban található):

```

HDVD (0) dupla keret
HSVS (1) szimpla keret
HSVD (2) vízszintesen szimpla, függőlegesen dupla
HDVS (3) vízszintesen dupla, függőlegesen szimpla
HNVN (4) üres keret

```


Visszatérési érték

Nincs.

Lásd még

aframe(), movframe(), <frame.h>

Példa

```
#include <frame.h>
```

```
frame(0,0,24,79,HSVS); /* teljes keret szimpla vonallal */
```

3.26. FREEBTAB

Összefoglalás

```
#include <bittab.h>
```

```
freebtab(pbt)
```

```
char *pbt;
```

a bittábla mutatója

Leírás

Felszabadítja a CREABTAB által lefoglalt dinamikus memóriaterületet.

Visszatérési érték

Nincs.

Lásd még

```
askbit(), clearbit(), creabtab(), clearbtabs(), setbit(),  
setbtabs()
```

Példa

Lásd a 3.14 CREABTAB c. pontnál.

char *name;	a file-név mintája
struct FC *pdta;	a munkapuffer mutatója
int fattr;	a file-attributum

Leírás

Megkeresi a névmintának és attributumnak megfelelő első file-t és feltölti a munkapuffert. A további (mintának megfelelő) file-ok a NEXTFILE függvénygel kérdezhetők le.

Az FC struktúra, a lehetséges file-attributumok és a file-dátumot és -időt kezelő makrók a következők (definíciójuk a <frstfile.h> header file-ban található):

Fájlattributumok:

R_ONLY	0x01	írásvédett
HIDDEN	0x02	rejtett
SYSTEM	0x04	rendszer
VOLABEL	0x08	lemez címke
SUBDIR	0x10	alkönyvtár
ARCHIVE	0x20	archivált

Az output FC struktúra (Disk Transfer Area)

```
struct FC
{
    char          f_res[21];      foglalt a DOS számára
    char          f_attr;         a file-attributum
    unsigned      f_time;         a file ideje
    unsigned      f_date;         a file dátuma
    long          f_size;         a file mérete byte-ban
    char          f_nam[13];      a file neve (0-val zárt)
};
```

Makrók az f_time és az f_date dekódolásához:

F_HOUR(t)	(t >> 11)	11-15 bit
F_MIN(t)	((t >> 5) & 0x3F)	5-10 bit
F_SEC(t)	((t & 0x1F) << 1)	0- 4 bit (mp/2)

F_YEAR(t)	((t >> 9) + 1980)	9-15 bit (1980-tól)
F_MONTH(t)	((t >> 5) & 0xF)	5- 8 bit
F_DAY(t)	(t & 0x1F)	0- 4 bit

Visszatérési érték

Ha nincs a mintának megfelelő file akkor 0, egyébként 1.

Lásd még

nextfile(), <frstfile.h>

Példa

```
#include <frstfile.h>
#define MAXFILES 512

/* az aktualis konyvtar .DAT tipusu file-jainak gyujtese */

char *files[MAXFILES]
int n;
struct FC buff;
char *fn;

n = 0;
if (frstfile("*.DAT",&buff,ARCHIVE) == 0)
{
    puttys("Nincs DAT file");
    return;
}

do
{
    fn = malloc(strlen(fc.f_nam)+1);
    strcpy(fn,fc.f_nam);
    files[n++] = fn;
}
while (nextfile(&fc));
```


3.29. GETCPOS

Összefoglalás

getcpos()

Leírás

Az aktuális kurzorpozíciót adja vissza tömörített formában.

Visszatérési érték

Az aktuális kurzorpozíció. A felső byte a sor, az alsó byte pedig az oszlop. A 0,0 pozíció a képernyő bal felső sarkát jelenti.

Lásd még

cupo(), setcpos()

Példa

```
int pos,row,col;

pos = getcpos();          /* sor es oszlop      */
row = pos >> 8;           /* sor            */
col = pos & 0xFF;         /* oszlop         */
```

3.30. GETDAT

Összefoglalás

getdat(date)

char *date;

puffer a dátum számára

Leírás

A puffert feltölti az aktuális dátummal, a következő formában: "Jan/31/1987". A puffer mérete minimálisan 12 byte.

Visszatérési érték

Nincs.

Lásd még

gettim()

Példa

```
char buffer[12];  
  
getdat(buffer);  
printf("A mai datum: %s",buffer);
```

3.31. GETDIR

Összefoglalás

char *getdir(drive,buffer)	
int drive;	a lemezmeghajtó sorszáma
char *buffer;	az output puffer

Leírás

A pufferbe beolvassa az aktuális könyvtár nevét a megadott lemezmeghajtón és visszaadja a könyvtárnevet lezáró '\'
(backslash) utáni mutatót. A lemezmeghajtó sorszáma:

0 - A:

1 - B:

.

.

Visszatérési érték

Ha a meghajtó sorszáma hibás akkor 0, egyébként a könyvtár nevét záró '\'
(backslash) utáni mutató a pufferen belül (az esetleges file-név kezdete).

Lásd még

currdisk(), getvol()

Példa

```
int i;
char buffer[80], *p,*getdir();

for (i = 0; i < 3; i++)
{
    if ((p = getdir(i,buffer)) == 0)
        printf("%c: nincs ilyen drive",i+'A');
    else
    {
        *(p-1) = '\\0';
        printf("A %c:-n a könyvtár: %s",i+'A',buffer);
    }
}
```

3.32. GETMODE

Összefoglalás

int getmode()

Leírás

Visszaadja az aktuális képernyőmódot.

Visszatérési érték

- | | |
|---|-------------------------------|
| 0 | 40x25 fekete-fehér |
| 1 | 40x25 színes |
| 2 | 80x25 fekete-fehér |
| 3 | 80x25 színes |
| 4 | 320x200 színes grafikus |
| 5 | 320x200 fekete-fehér grafikus |
| 6 | 640x200 fekete-fehér grafikus |

Lásd még

setmode()

Példa

```
/* indulas 80x25 szines modban */  
.  
setmode(4);           /* 320x200 szines grafikus mod */  
process();            /* grafikus programresz */  
setmode(3);           /* visszaallas az indulo modra */
```

3.33. GETPAGE

Összefoglalás

int getpage()

Leírás

Visszaadja az aktív képernyőlap sorszámát alfanumerikus üzemmódban.

Visszatérési érték

Az aktív képernyőlap sorszáma.

80x25 mód: 0-3

40x25 mód: 0-7

Lásd még

setpage()

Példa

```
if (getpage() > 0)  
{  
    puttys("Nem támogatott képernyőhasználat!");  
}
```



```
    exit(-1);  
}
```

3.34. GETTIM

Összefoglalás

```
gettim(time)  
    char *time;                puffer az idő számára
```

Leírás

A puffert feltölti az aktuális idővel, a következő formában: "12:23:56". A puffer mérete minimálisan 9 byte.

Visszatérési érték

Nincs.

Lásd még

```
getdat()
```

Példa

```
char buffer[9];  
  
gettim(buffer);  
printf("Indulas : %s\n",buffer);  
process();  
gettim(buffer);  
printf("Befejezes: %s\n",buffer);
```

3.35. GETTY

Összefoglalás

```
#include <kbget.h>
```


getty()

Leírás

Beolvassa a leütött billentyű ASCII kódját anélkül, hogy a karaktert kiírná a képernyőre. Ha a leütött billentyű kettős kódú (pl. funkcióbillentyűk, numerikus billentyűzet), akkor a GETTY függvény egy 0-ás értéket ad vissza, majd a második GETTY hívás adja meg a kettős kód felső értékét (scan-kód). Néhány gyakran használt billentyű ASCII kódja és a kettős kódú billentyűk felső értékeinek definiálása a <kbget.h> header file-ban található.

Visszatérési érték

A leütött billentyű kódja.

Lásd még

agetty(), getty(), kbget(), kbflush(), kbshifts(), kbstat(), <kbget.h>

Példa

```
#include <kbget.h>
int c;
int lin = 0;
int col = 0;

cupo(lin,col);
while ((c = getty()) != ESC)
    switch (c)
    {
        case ENTER:      doit();          break;
        case 'r':         readit();        break;
        case 'R':         readit();        break;
        case 's':         saveit();        break;
        case 'S':         saveit();        break;
        case 0:            switch(getty())
                           {
```



```

        case UP:
            if (--lin < 0)
                lin = 24;
            break;
        case DOWN:
            if (++lin > 24)
                lin = 0;
            break;
        case LEFT:
            if (--col < 0)
                col = 79;
            break;
        case RIGHT:
            if (++col > 79)
                col = 0;
            break;
        default:
            continue;
    }
    cupo(lin,col);
    break;
default:
    continue;
}

```

3.36. GETTYS

Összefoglalás

```

gettys(buffer,max)
    char *buffer;
    int max;

```

az output puffer
a beolvasható karakterek száma

Leírás

A billentyűzetről olvassa a pufferbe a karaktereket, mindaddig amíg ENTER-t vagy ESC-et nem kap.

Visszatérési érték

ESC (33 oktális, 27 decimális, 1B hexa), ha ESC zárta (ekkor az output puffer üres), egyébként 0 (ekkor az output pufferben a 0-val lezárt string található).

Lásd még

agetty(), agettys(), getty()

Példa

```
#define MAXNAME 50
```

```
char name[MAXNAME+1];
```

```
cupo(0,0);
```

```
puttys("Adja meg a file nevet: ");
```

```
if (gettys(name,MAXNAME) == '\33')  
    return(0);
```

```
.  
.  
.
```

3.37. GETVOL

Összefoglalás

```
char *getvol(drive,buffer)
```

```
int drive;
```

```
char *buffer;
```

a lemezmeghajtó sorszáma

a output puffer

Leírás

A megadott lemezmeghajtóban lévő lemez címkéjét betölti a pufferbe. A meghajtó sorszáma:

0 - A:

1 - B:

.
.

A puffer minimális mérete 12 byte.

Visszatérési érték

Ha meghajtó sorszáma rossz akkor 0, egyébként a puffer mutatója.

Példa

```
char volume[12];
int disk;

disk = currdisk();
getvol(disk, volume);
printf("A lemez címkeje: %s\n", volume);
```

3.38. INIMUSIC

Összefoglalás

```
#include <music.h>
```

```
inimusic(ptr)
    char *ptr;
```

a definiált zene byte-sorozata

Leírás

A megadott zenével bekapcsolja a háttérzenét. Ennek kikapcsolása az ENDMUSIC függvény hívásával érhető el. A zene definiálása a <music.h> header file-ban deklarált makrók segítségével történhet.

Visszatérési érték

Nincs.

Lásd még

endmusic(), music(), <music.h>

Példa

```
#include <music.h>
#include <window.h>
#include <setattr.h>
#include <frame.h>

char boci[] = {
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    Dd(3),ti(3),la(3),so(3),fa(2),la(2),
    so(3),fa(3),mi(3),re(3),dd(2),dd(2),
    pa(2),
    fine};

char szoveg[] = "Boci-boci tarka";

main()
{
    WINDTYPE sw;
    register int i,c;

    sw = windsave(8,31,14,47);      /* ablaknyitas */
    windfram(sw,HDVD,BLUE);        /* keretezes */
    for (i = 9,c = BLUE; i < 14; i++, c++)
    {
        cupo(i,32);
        aputtys(szoveg,c);
    }
    inimusic(boci);                /* hatterzene inditasa */
    while (!kbstat())              /* egy billentyu leuteseig */
    {
        movframe(1,9,32,13,46,GRAY); /* mozgatas folfele */
        cupo(13,32);                /* az also sor ujrairasa*/
        aputtys(szoveg,c);          /* mas szinnel */
        if (++c > WHITE)
            c = BLUE;
        delay(800);                 /* varakozas 8/10 s-ig */
    }
    kbget();                        /* billentyuzet puffer torlese */
    windclose(sw);                 /* ablak zarasa */
    endmusic();                    /* hatterzene leallitasa */
}
```


3.39. KBFLUSH

Összefoglalás

kbflush()

Leírás

Törli az előre beütött billentyűket a billentyűzet pufferéből.

Visszatérési érték

Nincs.

Lásd még

getty(), kbget(), kbstat()

Példa

```
putty("Usson le egy billentyut a folytatashoz...");  
kbflush();  
kbget();
```

3.40. KBGET

Összefoglalás

```
#include <kbget.h>
```

kbget()

Leírás

Beolvassa a leütött billentyű ASCII kódját és scan-kódját anélkül, hogy a karaktert kiírná a képernyőre. A visszatérési érték alsó byte-ja a billentyű ASCII kódja, a felső byte pedig a scan-kód. Ha a leütött billentyű kettős kódú (pl. funkció-billentyűk, numerikus billentyűzet), akkor az ASCII kód (alsó

byte) 0 és a felső byte azonosítja a billentyűt. A kettős kódú billentyűk felső értékeinek definiálása a <kbget.h> header file-ban található.

Visszatérési érték

A leütött billentyű ASCII kódja (alsó byte) és scan-kódja (felső byte).

Lásd még

getty(), kbflush(), kbshifts(), kbstat(), <kbget.h>

Példa

```
int code, scan, charcode;

code = kbget();           /* ASCII es scan-kod    */
charcode = code & 0xFF;   /* ASCII kod            */
scan = code >> 8;        /* scan-kod             */
```

3.41. KBSHIFTS

Összefoglalás

```
#include <kbshifts.h>
```

```
kbshifts()
```

Leírás

Visszaadja a billentyűzet shift-státuszát, tehát hogy melyik shift-billentyű (bal shift, jobb shift, Ctrl...) van lenyomva vagy bekapcsolt állapotban. A visszatérési érték bitmaszkjainak definíciója a <kbshifts.h> header file-ban található.

Visszatérési érték

Az aktuális shift-státusz.

Lásd még

kbget(), kbstat(), <kbshifts.h>

Példa

```
#include <kbshifts.h>
```

```
if (kbshifts() & NUMLOCK)
{
    putty("A kurzorbillentyű Num_lock eseten nem működnek!");
    while (kbshifts() & NUMLOCK)
        ;
}
```

3.42. KBSTAT

Összefoglalás

```
#include <kbget.h>
```

```
kbstat()
```

Leírás

Megadja, hogy volt-e leütött billentyű.

Visszatérési érték

Ha nem volt leütött billentyű akkor 0, egyébként a leütött billentyű ASCII és scan-kódja (a KBGET mintájára, viszont a KBSTAT a kódpárt nem törli a pufferből, tehát az a KBGET-tel még beolvasható).

Lásd még

getty(), kbflush(), kbget(), kbshifts(), <kbget.h>

Példa

```
int c;
```



```

while ( (c = kbstat()) == 0)
    ;
switch (c & 0xFF)      /* csak az ASCII kodot vizsgáljuk */
{
    case ENTER:
        kbget();      /* tenyleges beolvasas */
        .
        .

```

3.43. LINDOUBLE

Összefoglalás

```
#include <setattr.h>
```

```
lindouble(lin,col,len,attr)
```

int lin;	a kezdő sor a képernyőn
int col;	az oszlop
int len;	a vonal hossza
int attr;	a kívánt szín

Leírás

A képernyőn lévő keretelemek közé beillesztve egy dupla vízszintes vonalat rajzol alfanumerikus üzemmódban. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

```

aframe(), coldouble(), colsimple(), frame(), linsimple(),
<frame.h>, <setattr.h>

```

Példa

```

#include <setattr.h>
#include <frame.h>

```



```
aframe(0,10,20,40,HDVD,GRAY); /* keretezes */
lindouble(10,10,31,GRAY); /* a keret ketteosztasa */
```

3.44. LINSIMPLE

Összefoglalás

```
#include <setattr.h>
```

```
linsimple(lin,col,len,attr)
    int lin;          a kezdő sor
    int col;          az oszlop
    int len;          a vonal hossza
    int attr;         a kívánt szín
```

Leírás

A képernyőn lévő keretelemek közé beillesztve egy vízszintes vonalat rajzol alfanumerikus üzemmódban. A lehetséges színértékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

```
aframe(), colsimple(), coldouble(), frame(), lindouble(),
<frame.h>, <setattr.h>
```

Példa

```
#include <setattr.h>
#include <frame.h>

aframe(0,10,20,40,HSVD,GRAY); /* keretezes */
linsimple(10,10,31,GRAY); /* a keret ketteosztasa */
```


3.45. MOVESCR

Összefoglalás

<code>movescr(ulr,ulc,lrr,lrc,new_ulr,new_ulc)</code>	
<code>int ulr;</code>	a keret felső sora
<code>int ulc;</code>	a keret bal oszlopa
<code>int lrr;</code>	a keret alsó sora
<code>int lrc;</code>	a keret jobb oszlopa
<code>int new_ulr;</code>	a keret új felső sora
<code>int new_ulc;</code>	a keret új bal oszlopa

Leírás

A megadott képernyőterületet átmásolja a 'new_ulr' és 'new_ulc' által meghatározott területre.

Visszatérési érték

Nincs.

Lásd még

`aframe()`, `frame()`, `movframe()`, `windmove()`

Példa

```
erase();  
puttys("Első sor");  
puttys("Második sor");  
puttys("Harmadik sor");  
movescr(0,0,2,20,0,40); /* a szöveg másolása a 40. oszlophoz */
```

3.46. MOVFRAME

Összefoglalás

```
#include <setattr.h>
```


movframe(lines,ulr,ulc,lrr,lrc,attr)	
int lines;	a mozgatandó sorok száma
int ulr;	a keret felső sora
int ulc;	a keret bal oszlopa
int lrr;	a keret alsó sora
int lrc;	a keret jobb oszlopa
int attr;	a kívánt színezés

Leírás

A megadott képernyőrészt felfelé vagy lefelé mozgatja. Ha a mozgatandó sorok száma 0, akkor törli a megadott képernyőrészt. Ha a mozgatandó sorok száma nagyobb mint 0, akkor a képernyőrészt ennyivel felfelé mozgatja, ha a mozgatandó sorok száma kisebb mint 0, akkor ennek abszolút értékével mozgat lefelé. Az új sorok színe az 'attr' lesz. A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

aframe(), frame(), movescr(), windmove(), <setattr.h>

Példa

```
char *sorok[] = {"Elso", "Masodik", "Harmadik", "Negyedik"};
int i;
```

```
for (i = 0; i < 4; i++)
{
    cupo(i,0);
    aputtysn(sorok[i],GRAY,8);
}
```

```
i = 0;
```

```
/* utolso kiirt soron allunk */
```

```
while (!kbstat())                /* egy billentyu leuteseig */
{
```



```

        movframe(1,0,0,3,7,GRAY);      /* mozgatas felfele */
        aputtysn(sorok[i],GRAY,8);      /* pozicio nem valtozik */
        if (++i >= 4)
            i = 0;
    }
    kbget();                             /* leutott billentyu torlese */

```

3.47. MUSIC

Összefoglalás

```
#include <music.h>
```

```
music(ptr)
    char *ptr;           a definiált zene byte-sorozata
```

Leírás

Lejátssza a megadott zenét. A zene definiálása a <music.h> header file-ban deklarált makrók segítségével történhet.

Visszatérési érték

Nincs.

Lásd még

endmusic(), inimusic(), <music.h>

Példa

```
#include <music.h>
```

```

char boci[] = (
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    Dd(3),ti(3),la(3),so(3),fa(2),la(2),
    so(3),fa(3),mi(3),re(3),dd(2),dd(2),
    pa(2),
    fine);

```



```
.  
.  
music(boci);  
.  
.
```

3.48. NEXTFILE

Összefoglalás

```
#include <frstfile.h>
```

```
nextfile(pdta)
```

```
    struct FC *pdta;
```

a munkapuffer mutatója

Leírás

A pufferbe tölti a FRSTFILE által már definiált mintának megfelelő következő file adatait. A 'pdta' meg kell, hogy egyezzen a FRSTFILE-nak adott második paraméterrel.

Az FC struktúra, a lehetséges file-attributumok és a file-dátumot és -időt kezelő makrók leírása a 3.28 FRSTFILE c. pontnál és a <frstfile.h> header file-ban található.

Visszatérési érték

Ha nincs több, a mintának megfelelő file akkor 0, egyébként 1.

Lásd még

```
frstfile(), <frstfile.h>
```

Példa

Lásd a 3.28 FRSTFILE c. pontnál.

3.49. NOSOUND

Összefoglalás

nosound()

Leírás

Kikapcsolja a SOUND által bekapcsolt hangszórót.

Visszatérési érték

Nincs.

Lásd még

delay(), endmusic(), inimusic(), music(), sound()

Példa

```
sound(500);      /* a hangszoro bekapcsolasa 500 Hz hangon */
delay(3000);     /* 3 masodpercig hallhato */
nosound();       /* a hangszoro kikapcsolasa */
```

3.50. PARSNAME

Összefoglalás

```
#include <parsname.h>
```

```
char *parsname(defdrive,path,fullspec)
    int defdrive;          a default lemezmeghajtó sorszáma
    char *path;            a vizsgálandó név
    char *fullspec;        az output puffer
```

Leírás

Megvizsgálja az átadott nevet és az output pufferbe tölti a teljes file-specifikációt (lemez:\könyvtár\file-név). Ha a

vizsgálandó név egy könyvtár neve, akkor a ".*.*" file-nevet fűzi a specifikációhoz.

Visszatérési érték

Ha a vizsgálandó név hibás akkor 0, egyébként a file-név mutatója az output pufferen belül.

Lásd még

currdisk(), parspath(), <parsname.h>

Példa

```
#include <parsname.h>

main(argc,argv)
int argc;
char *argv[];
{
    char inspec[80];
    char outdir[80];
    char *iname;
    char *outname;

    if (argc < 2)
    {
        puts("Parameterek: input-file output-konyvtar");
        exit(0);
    }

    if ((iname = parspath(currdisk(),argv[1],inspec)) == 0)
        exit(-1);
    if ((outname = parspath(currdisk(),argv[1],outdir)) == 0)
        exit(-1);
    printf("Mehet a %s file-ok masolasa a %s könyvtarba?",
        inspec,outdir);
    .
    .
}
```


3.51. PARSPATH

Összefoglalás

```
#include <parsname.h>
```

```
char *parspath(defdrive,path,fullpath)
    int defdrive;           a default lemezmeghajtó sorszáma
    char *path;             a vizsgálandó név
    char *fullspec;         az output puffer
```

Leírás

Megvizsgálja az átadott nevet és az output pufferbe tölti a teljes könyvtár-specifikációt (lemez:\könyvtár\).

Visszatérési érték

Ha a vizsgálandó név hibás akkor 0, egyébként a file-név helyének mutatója az output pufferen belül (a záró '\' utáni pozíció).

Lásd még

```
currdisk(), parsname(), <parsname.h>
```

Példa

Lásd a 3.50 PARNAME c. pontnál.

3.52. PRINHELP

Összefoglalás

```
prinhelp(helptxt,chelp)
    char *helptxt;          a help szövege
    int chelp;              a help színe
```


Leírás

Dinamikus ablakot nyit a képernyő közepén és ebben megjeleníti a help szövegét a megadott színnel. A help szövege maximum 23 soros lehet, és egyik sor hossza sem haladhatja meg a 78 (40x25-ös módban a 38) karaktert. A kiírás után a kurzorbillentyűkkel a help ablaka elmozgatható. ESC hatására visszaáll az eredeti képernyő.

Visszatérési érték

Nincs.

Példa

```
#include <kbget.h>
#include <setattr.h>

char htxt[] = "F1:  help\r\n\
UP:   move cursor up\r\n\
DOWN: move cursor down\r\n\
LEFT: move cursor left\r\n\
RIGHT: move cursor right\r\n\
ESC:  exit";

int c; int lin = 0; int col = 0;

while ( (c = getty()) != ESC)
{
    if (c != 0)
        continue;
    switch (getty())
    {
        case UP:
            if (--lin < 0)
                lin = 24;
            break;
        case DOWN:
            if (++lin > 24)
                lin = 0;
            break;
```



```

case LEFT:
    if (--col < 0)
        col = 79;
    break;
case RIGHT:
    if (++col > 79)
        col = 0;
    break;
case F1:
    printhelp(htxt,COLOR(BLACK, GREEN));
default:
    continue;
cupo(lin,col);
}

```

3.53. PROMPT

Összefoglalás

prompt(promptstr,args,maxargs)

char *promptstr;

char **args;

int maxargs;

a bejelentkező szöveg

az ARGV-nek megfelelő mutatótömb

a beolvasható paraméterek száma

Leírás

Kiírja a bejelentkező szöveget, majd a felhasználó által beütött parancssort az ARGV-ARGV programparamétereknek megfelelő módon feltölti. Használata lehetővé teszi egy programnak a betöltése utáni paraméterezését, tehát a tényleges feldolgozó rész indulásakor a programot tartalmazó lemeznek már nem kell a meghajtóban lennie.

Visszatérési érték

A programparaméterek száma.

Példa

```
#define MAXARGS 10
```

```
main(argc,argv)
int argc;
char **argv;
{
char *args[MAXARGS+1];

if (argc <= 1)
    argc = prompt("MYPROMPT>", (argv = args), MAXARGS);

while (argc--)
    puts(*++argv);
}
```

3.54. PUTTY

Összefoglalás

```
putty(c)
char c;           a kiírandó karakter
```

Leirás

A karaktert kiírja az aktuális pozícióra, majd a kurzort eggyel jobbra lépteti.

Visszatérési érték

Nincs.

Lásd még

`aputty()`, `awrchar()`, `puttys()`, `wrchar()`

3.55. PUTTYS

Összefoglalás

```
puttys(str)
char *str;           a kiírandó string
```

Leirás

A stringet kiírja az aktuális pozíciótól kezdve. Kiírás után a kurzor a kiírt string mögé áll.

Visszatérési érték

Nincs.

Lásd még

aputty(), awrchar(), aputtys(), aputtysn(), putty(), wrchar()

3.56. RDCHAR

Összefoglalás

rdchar()

Leirás

A képernyő aktuális pozíciójáról beolvassa az ott lévő karaktert és annak színét. A visszatérési érték alsó byte-ja a karakter ASCII kódja, a felső byte pedig a szín.

Visszatérési érték

Az aktuális pozíción lévő karakter kódja és színe.

Példa

```
int code;
```



```
int ch;  
int attr;
```

```
code = rdchar();  
ch = code & 0xFF;  
attr = code >> 8;
```

3.57. RS232INIT

Összefoglalás

```
#include <rs232.h>
```

```
int rs232init(baud,parity,stopbit,wlength,port)  
    int baud;                az átviteli sebesség (baud rate)  
    int parity;              a paritás (Odd-Even-None)  
    int stopbit;             a stop-bit (1-2)  
    int wlength;             a szó hossz (7-8 bit)  
    int port;                a kommunikációs vonal (0-1)
```

Leírás

Inicializálja a megadott kommunikációs vonalat. A lehetséges paraméterek leírása a <rs232.h> header file-ban található.

Visszatérési érték

Ha a paraméterek között van hibás akkor -1, egyébként 0.

Lásd még

rs232rece(), rs232send(), rs232stat(), <rs232.h>

3.58. RS232RECE

Összefoglalás

```
#include <rs232.h>
```



```
int rs232rece(port)
```

```
int port;
```

a kommunikációs vonal (0-1)

Leírás

A megadott kommunikációs vonalról beolvas egy byte-ot.

Visszatérési érték

A vonalról olvasott byte. Ha hiba történt az átvitelben akkor a felső byte nem nulla, ekkor a felső byte a hiba kódját tartalmazza. A hibakódok leírása a <rs232.h> header file-ban található.

Lásd még

rs232init(), rs232send(), rs232stat(), <rs232.h>

3.59. RS232SEND

Összefoglalás

```
#include <rs232.h>
```

```
int rs232send(byte, port)
```

```
int byte;
```

```
int port;
```

a küldendő byte

a kommunikációs vonal (0-1)

Leírás

A megadott vonalon elküldi a megadott byte-ot.

Visszatérési érték

Nincs.

Lásd még

rs232init(), rs232rece(), rs232stat(), <rs232.h>

3.60. RS232STAT

Összefoglalás

```
#include <rs232.h>
```

```
int rs232stat(port)
```

```
    int port;
```

a kommunikációs vonal (0-1)

Leírás

Visszaadja a megadott vonal státuszát. A státusz bit-maszkjainak leírása a <rs232.h> header file-ban található.

Visszatérési érték

A kommunikációs vonal státusza.

Lásd még

rs232init(), rs232rece(), rs232send(), <rs232.h>

3.61. SELECT

Összefoglalás

```
#include <select.h>
```

```
int select(itnum, itlen, scrfp, title, cbase, chighl, workfp, letters)
```

```
    int itnum;          a tételek száma
```

```
    int itlen;          a tételek maximális hossza
```

```
    FVP scrfp;          a tételkiíró függvény mutatója
```

```
    char *title;        az ablakba kerülő cím
```

```
    int cbase;          az alapszín
```

```
    int chighl;          a kiemelt szín
```

```
    FVP workfp;         az ENTER-re végrehajtandó fv. mutatója
```

```
    char *letters;      a kiválasztó betűk sorozata
```


Leírás

A SELECT valamely tételtömb egy elemének kiválasztását (és opcionálisan annak manipulálását) ablakos képernyőkezeléssel megvalósító rutin.

A SELECT funkciói:

A képernyő jobb felső sarkához igazítva egy szükséges méretű dinamikus ablakot nyit.

Az ablakba kiíratja a tételeket.

Az UP, DOWN, PGUP, PGDN, HOME, END kurzorbillentyűk segítségével lehetőséget ad egyétel kiválasztására.

Az utolsó paraméterként megadott betűsorozat megfelelő elemének leütésével lehetőséget ad a betűnek megfelelőétel kiválasztására.

A CTR_LEFT, CTR_RIGHT, CTR_PGUP és a CTR_PGDN kontroll-kurzorbillentyűk segítségével lehetőséget ad az ablak mozgatására.

ESC leütése esetén visszaállítja az eredeti képernyőt és -1 értéket ad vissza.

ENTER leütése esetén

- ha a 'workfp' függvénymutató 0 értékű, akkor visszaállítja az eredeti képernyőt és visszaadja a kiválasztottétel indexét,
- ha a 'workfp' függvénymutató nem 0, akkor végrehajtja a mutatóval azonosított függvényt a következő paraméterekkel:
 - az aktuálisétel indexe,
 - az aktuális sor a képernyőn,
 - az aktuális oszlop a képernyőn.

Visszatérési érték

- 2 hiba esetén (keves memória),

-1 ESC leütése esetén,
a kiválasztott tétel indexe, ha a 'workfp' függvénymutató 0
értékű.

Lásd még

selfile(), xselect(), <select.h>

Példa

```
#include <select.h>
#include <setattr.h>

char *tttext[] = {
    "Integer",
    "Long",
    "Float",
    "Double"
};

.
.
.
{
int ttoscreen();
int typ;

if ((typ=select(4,7,ttoscreen,"Tipusok",GRAY,WHITE,0,"ILFD")) < 0)
    return;
switch(typ)
{
.
.
}

static int ttoscreen(act,lin,col)
int act,lin,col;
{
    cupo(lin,col);
    aputtysn(tttext[act],GRAY,7);
}
```


3.62. SELFIE

Összefoglalás

#include <selfie.h>

```
int selfie(spec,new_str,is_for_write,cbase,chighl,buffer)
char *spec;           a file-minta
char *new_str;        írási mód esetén az új tétel jelölése
int is_for_write;     a file-választás írásra történjen?
int cbase;            az alapszín
int chighl;           a kiemelt szín
char *buffer;         az output puffer
```

Leírás

A képernyő jobb felső sarkában dinamikus ablakot nyit.

A létező file-ok neveit rendezett sorban kiírja az ablakba

- ha a 'spec' minta fix típust tartalmaz (pl.: *.DAT), akkor csak a file-nevek kerülnek kiírásra és a visszaadott név is ezt a típust fogja tartalmazni.
- ha a 'spec' minta nem tartalmaz fix típust (pl.: *.D??), akkor a kiírás file-típussal együtt történik.

Az UP, DOWN, PGUP, PGDN, HOME, END kurzorbillentyűk segítségével lehetőséget ad egy file-név kiválasztására.

A CTR_LEFT, CTR_RIGHT, CTR_PGUP és a CTR_PGDN kontroll-kurzorbillentyűk segítségével lehetőséget ad az ablak mozgatására.

ESC leütése esetén visszaállítja az eredeti képernyőt és SF_ESC értéket ad vissza.

ENTER leütése esetén

- ha a kiválasztás olvasásra történt, vagy egy már létező file a kiválasztott (írásra kiválasztás esetén), akkor visszaállítja az eredeti képernyőt, visszaadja az SF_SEL értéket, és az output pufferbe a kiválasztott file teljes neve kerül,

- írásra kiválasztásnál az új file tétel választása esetén egy külön kis ablakban lehet megadni a létrehozandó file nevét:
 - ESC leütésére visszaáll az eredeti képernyő és a vissza térési érték SF_ESC,
 - értelmes file-név megadása esetén visszaállítja az eredeti képernyőt, visszaadja az SF_NEW értéket és az output pufferbe a megadott file teljes neve kerül.

Visszatérési érték

SF_ERR - hiba esetén (nemlétező file),
 SF_ESC - ESC leütése (kiválasztás visszautasítása),
 SF_SEL - létező file kiválasztásakor,
 SF_NEW - új file megadásakor.

Lásd még

select(), xselect(), <selfile.h>

Példa

```
#include <selfile.h>

/* az aktualis konyvtar .DAT tipusu file-jaibol valo valasztas */

char fname[80];

res = selfile("*.DAT",0,SF_READ,GRAY,YELLOW,fname);
switch (res)
{
  case SF_ERR:
    puttys("Nincs ilyen file!");
    return(-1);
  case SF_ESC:
    return(0);
  case SF_SEL:
    process(fname);
    .
    .
}
}
```


3.63. SETATTR

Összefoglalás

```
#include <setattr.h>
```

```
setattr(length,attr)
```

```
    int length;
```

```
    int attr;
```

az átállítandó mező hossza

az új szín

Leírás

Az aktuális pozíciótól kezdve a megadott számú pozíció színezését állítja be a karakterek változatlanul hagyásával. A lehetséges színek és makrók (definíciójuk a <setattr.h> header file-ban található):

Színek:

BLACK	0	(fekete)
BLUE	1	(kék)
GREEN	2	(zöld)
CYAN	3	(világoskék)
RED	4	(piros)
MAGN	5	(lila)
BROWN	6	(barna)
GRAY	7	(világosszürke)

Erős színek (az előbbieket intenzív változatai):

DGRAY	INTENSIV(BLACK)
LBLUE	INTENSIV(BLUE)
LGREEN	INTENSIV(GREEN)
LCYAN	INTENSIV(CYAN)
LRED	INTENSIV(RED)
LMAGN	INTENSIV(MAGN)
YELLOW	INTENSIV(BROWN)
WHITE	INTENSIV(GRAY)

Egyéb karakterattributum makrók:

BLIND(c)	villogó szín
INTENSIV(c)	erős színváltozat (lásd fenn)
BACKGR(c)	háttérszín
COLOR(f,b)	előtér- és háttérszín

Visszatérési érték

Nincs.

Lásd még

aputtys(), awrchar(), <setattr.h>

Példa

```
#include <setattr.h>
#include <kbget.h>
main()
{
    int c;
    erase();
    aputtys(" Bal ",WHITE);
    aputtys("Jobb ",GRAY);
    while ((c = getty()) != ESC)
    {
        if ( (c = getty()) != LEFT && c != RIGHT)
            continue;
        cupo(0,0);
        setattr(10,GRAY);
        if (c == RIGHT)
            cupo(0,5);
        setattr(5,WHITE);
    }
}
```


3.64. SETBIT

Összefoglalás

```
#include <bittab.h>
```

```
setbit(lin,col,pbt,size)
```

```
int lin,col;
```

```
char *pbt;
```

```
int size;
```

a beállítandó bit koordinátái

a bittábla mutatója

a bittábla mérete

Leírás

A bittábla megadott bitjét igazra állítja.

Visszatérési érték

Nincs.

Lásd még

askbit(), clearbit(), clearbtabs(), creabtabs(), freebtabs(),
setbtabs()

Példa

Lásd a 3.14 CREABTAB c. pontnál.

3.65. SETBTAB

Összefoglalás

```
#include <bittab.h>
```

```
setbtabs(pbt,size)
```

```
char *pbt;
```

```
int size;
```

a bittábla mutatója

a bittábla mérete

Leirás

A CREATAB rutinnal létrehozott bittábla minden bitjét igazra állítja.

Visszatérési érték

Nincs.

Lásd még

askbit(), clearbit(), clearbtab(), creabtab(), freebtab(), setbit()

Példa

Lásd a 3.14 CREABTAB c. pontnál.

3.66. SETCPOS

Összefoglalás

```
setcpos(pos)
    int pos;           a kívánt pozíció
```

Leirás

A kurzort a megadott pozícióra állítja. Az átadott paraméter felső byte-ja tartalmazza a kívánt sort, az alsó byte pedig a kívánt oszlopot. A képernyő bal felső sarka jelenti a 0,0 pozíciót.

Visszatérési érték

Nincs.

Lásd még

cupo(), getcpos()

Példa

```
int savepos;

setcpos(0x1800);          /* also sor elso pozicio */
setcpos( (5 << 8) | 10); /* 5. sor 10. oszlop */

savepos = getcpos();      /* kurzorpozicio mentese */
process();
setcpos(savepos);         /* kurzorpozicio visszamentese */
```

3.67. SETMODE

Összefoglalás

```
setmode(pos)              a kívánt mód
    int mode;
```

Leírás

Beállítja a képernyőmódot.

Lehetséges értékek:

- | | |
|---|-------------------------------|
| 0 | 40x25 fekete-fehér |
| 1 | 40x25 színes |
| 2 | 80x25 fekete-fehér |
| 3 | 80x25 színes |
| 4 | 320x200 színes grafikus |
| 5 | 320x200 fekete-fehér grafikus |
| 6 | 640x200 fekete-fehér grafikus |

Visszatérési érték

Nincs.

Lásd még

getmode()

Példa

```
/* indulas 80x25 szines modban */  
.  
setmode(4);           /* 320x200 szines grafikus mod */  
process();            /* grafikus programresz */  
setmode(3);           /* visszaallas az indulo modra */
```

3.68. SETPAGE

Összefoglalás

```
setpage(page)  
    int page;           a kívánt képernyőlap
```

Leírás

Beállítja a kívánt képernyőlapot alfanumerikus üzemmódban. A
'page' lehetséges értékei:
 0-3 80x25 módban,
 0-7 40x25 módban

Visszatérési érték

Nincs.

Lásd még

```
getpage()
```

Példa

```
setpage(1);           /* atteres a masodik lapra */  
process();            /* nem használja az első lapot */  
setpage(0);           /* visszateres az első lapra */
```


3.69. SOUND

Összefoglalás

sound(herz)
int herz; a kívánt hangmagasság (Hz)

Leirás

A megadott hangmagasságban megszólaltatja a hangszórót.

Visszatérési érték

Nincs.

Lásd még

delay(), endmusic(), inimusic(), music(), nosound()

Példa

```
sound(500);      /* a hangszoro bekapcsolasa 500 Hz hangon */
delay(3000);     /* 3 masodpercig hallhato */
nosound();       /* a hangszoro kikapcsolasa */
```

3.70. STRDUP

Összefoglalás

```
#include <aztec.h>
```

```
char *strdup(str)
char *str;           a másolandó string
```

Leirás

A Microsoft C-ben szabványos, de az Aztec C-ből hiányzó függvény, ezért kompatibilitási okból szerepel az ACEx

könyvtárban! Használata esetén az Aztec C compilert a -dAZTEC opcióval kell indítani.

A megadott string számára memóriaterületet foglal le, majd a stringet átmásolja az új területre.

Visszatérési érték

Ha nincs elég memória akkor 0, egyébként az átmásolt új string mutatója.

Példa

```
#include <aztec.h>

char *sorok[3];

sorok[0] = strdup("Első sor");
sorok[1] = strdup("Második sor");
sorok[2] = strdup("Harmadik sor");
```

3.71. STRLWR

Összefoglalás

```
#include <aztec.h>

char *strlwr(str)
    char *str;                a konvertálandó string
```

Leírás

A Microsoft C-ben szabványos, de az Aztec C-ből hiányzó függvény, ezért kompatibilitási okból szerepel az ACEX könyvtárban! Használata esetén az Aztec C compilert a -dAZTEC opcióval kell indítani.

A megadott stringben az összes nagybetűt kisbetűre konvertálja. A többi karakter változatlan marad.

Visszatérési érték

Az átadott string.

Lásd még

strupr()

Példa

```
#include <aztec.h>
```

```
char *p;
```

```
p = strdup("AbcDEf");
```

```
p = strlwr(p);
```

```
/* eredmény: "abcdef" */
```

3.72. STRUPR

Összefoglalás

```
#include <aztec.h>
```

```
char *strupr(str)
```

```
char *str;
```

a konvertálandó string

Leírás

A Microsoft C-ben szabványos, de az Aztec C-ből hiányzó függvény, ezért kompatibilitási okból szerepel az ACEX könyvtárban! Használata esetén az Aztec C compilert a -dAZTEC opcióval kell indítani.

A megadott stringben az összes kisbetűt nagybetűre konvertálja. A többi karakter változatlan marad.

Visszatérési érték

Az átadott string.

Lásd még

strlwr()

Példa

```
#include <aztec.h>
```

```
char *p;
```

```
p = strdup("AbcDEf");
```

```
p = strupr(p); /* eredmény: "ABCDEF" */
```

3.73. WINDCLOSE

Összefoglalás

```
#include <window.h>
```

```
windclose(screenbuf)
```

```
WINDTYPE screenbuf;
```

az ablak puffere

Leírás

Visszaállítja az eredeti képernyőt és felszabadítja a lekötött memóriaterületet. Hatása megegyezik a WINDREST és WINDFREE függvények egymás utáni hívásával.

Visszatérési érték

Nincs.

Lásd még

windfram(), windfree(), windmove(), windrest(), windsave(),
<window.h>

Példa

Lásd a 3.78 WINDSAVE c. pontnál.

3.74. WINDFRAM

Összefoglalás

```
#include <window.h>
#include <frame.h>
#include <setattr.h>
```

```
windfram(screenbuf, frametyp, attr)
    WINDTYPE screenbuf;      az ablak puffere
    int      frametyp;       a kívánt kerettípus
    int      attr;           az ablak színezése
```

Leírás

Egy WINDSAVE által megnyitott ablak törlését és "táguló" keretezését végzi. A lehetséges kerettípusok (definíciójuk a <frame.h> header file-ban található):

- HDVD (0) dupla keret
- HSVS (1) szimpla keret
- HSVD (2) vízszintesen szimpla, függőlegesen dupla
- HDVS (3) vízszintesen dupla, függőlegesen szimpla
- HNVN (4) üres keret

A lehetséges színezési értékek leírása a <setattr.h> header file-ban és a 3.63 SETATTR c. pontban található.

Visszatérési érték

Nincs.

Lásd még

frame(), movframe(), windmove(), windrest(), windsave(),
<frame.h>, <setattr.h>, <window.h>

Példa

Lásd a 3.78 WINDSAVE c. pontnál.

3.75. WINDFREE

Összefoglalás

```
#include <window.h>
```

```
windfree(screenbuf)
```

```
    WINDTYPE screenbuf;
```

az ablak puffere

Leírás

Felzabadítja a WINDSAVE által az ablak számára lekötött memóriaterületet. (A WINDREST hívása után használandó)

Visszatérési érték

Nincs.

Lásd még

```
windclose(), windfram(), windmove(), windrest(), windsave(),  
<window.h>
```

Példa

Lásd a 3.78 WINDSAVE c. pontnál.

3.76. WINDMOVE

Összefoglalás

```
#include <window.h>
```

```
int windmove(screenbuf,direction)
```

```
    WINDTYPE screenbuf;
```

az ablak puffere

```
    int direction;
```

a mozgás iránya

Leírás

Egy a WINDSAVE által megnyitott ablakot mozgat el egy pozícióval a megadott irányba.

A lehetséges mozgatósi irányok (definíciójuk a <window.h> header file-ban található):

WM_UP	0	felfelé
WM_LEFT	1	balra
WM_DOWN	2	lefelé
WM_RIGHT	3	jobbra

Visszatérési érték

Ha a mozgatás nem lehetséges akkor -1, egyébként 0.

Lásd még

movframe(), windfram(), windrest(), windsave(), <window.h>

Példa

Lásd a 3.78 WINDSAVE c. pontnál.

3.77. WINDREST

Összefoglalás

```
#include <window.h>
```

```
windrest(screenbuf)
```

```
WINDTYPE screenbuf;
```

az ablak puffere

Leírás

Visszaállítja az ablak eredeti tartalmát anélkül, hogy az ablak puffert törölné.

Visszatérési érték

Nincs.

Lásd még

windclose(), windfram(), windfree(), windmove(), windsave(),
<window.h>

Példa

Lásd a 3.78 WINDSAVE c. pontnál.

3.78. WINDSAVE

Összefoglalás

```
#include <window.h>
```

```
WINDTYPE windsave(ulr,ulc,lrr,lrc)
    int ulr;           az ablak felső sora
    int ulc;           az ablak bal oszlopa
    int lrr;           az ablak alsó sora
    int lrc;           az ablak jobb oszlopa
```

Leírás

Lementi a kijelölt terület eredeti tartalmát és ablakot nyit erre a területre.

Visszatérési érték

Ha kevés a memória akkor 0, egyébként a lementett terület dinamikus pufferének címe.

Lásd még

windclose(), windfram(), windfree(), windmove(), windrest(),
<window.h>

Példa

```
#include <window.h>
#include <setattr.h>
#include <frame.h>
#include <kbget.h>
```

```
WINDTYPE ws; int c;
```

```
if ((ws = windsave(5,20,14,60)) == 0)           /* ablaknyitas */
{
    puttys("Tul keves a memoria!");
    return;
}
```

```
windfram(ws,HDVD,GRAY);           /* alapszinu kettos keretezes */
```

```
/* a 4,21,13,59 terület feltoltese */
```

```
while ((c = getty()) != ESC)
```

```
{
    if (c == 0)
    {
        switch (getty())           /* scan-kod */
        {
            case LEFT:
                windmove(ws,WMLLEFT); break;
            case RIGHT:
                windmove(ws,WMLRIGHT); break;
            case UP:
                windmove(ws,WMLUP); break;
            case DOWN:
                windmove(ws,WMLDOWN); break;
        }
    }
}
```

```
windrest(ws);           /* az eredeti kepernyo visszaallitasa */
windfree(ws);           /* ablak puffereinek felszabaditasa */
```


3.79. WRCHAR

Összefoglalás

wrchar(num, ch)	
unsigned num;	a karakter ismétlési száma
char ch;	a kiírandó karakter

Leírás

Kiírja a karaktert a képernyőre az aktuális pozíciótól kezdve annyiszor, amennyi az ismétlési tényező. A kurzor pozíciója nem változik.

Visszatérési érték

Nincs.

Lásd még

aputty(), awrchar(), putty()

Példa

```
cupo(23,0);  
wrchar(160,' '); /* utolso ket sor torlese */
```

3.80. XSELECT

Összefoglalás

```
#include <xselect.h>  
#include <setattr.h>  
#include <>window.h>  
  
int xselect(xs, screenbuf)  
    struct XSELECT xs;  
    WINDTYPE screenbuf;
```

a definíciós struktúra mutatója
a WINDSAVE-vel lementett ablak

Leírás

Az XSELECT rutin egy magas szinten automatizált, teljesképernyős eljárás tömbök, tételsorozatok manipulálására. Az XSELECT használata első pillantásra nem egyszerű, de gazdag szolgáltatásai miatt használatát érdemes elsajátítani.

Az XSELECT segítségével egy tömb elemét kiválaszthatjuk, oda-lapozhatunk a képernyőn el nem férő elemekhez, módosíthatjuk a kiválasztott elemet, új elemet szűrhatunk be a tömb elemei közé, eltörölhetünk egy tömbelemet, végrehajthatjuk a funkció billentyűkhöz rendelt C függvényeket, stb... Mindezt az XSELECT értelmes billentyűzetkezelés mellett, a felhasználó számára áttekinthetően valósítja meg, megkímélve a programozót a kurzor vezérlés, a lapozás, a billentyűértelmezés fárasztó és sok hibalehetőséget magában foglaló programozásától. Mivel az XSELECT az egyes funkciókat (egy tétel kiírása, egy tétel kiemelése, beszúrás, törlés, stb.) a felhasználó által megírt függvények hívásával végzi, az XSELECT megfelelően rugalmas, és használatával a teljesképernyős adatkezelés sok válfaja megvalósítható.

A programozó által írt függvényeket az XSELECT az alábbi pontokban leírt (szükséges és elégséges) paraméterekkel hívja meg, tehát a függvényeknek kezelniük kell ezeket. Az XSELECT legtöbb esetben (lásd később) figyeli a programozó függvényeinek visszatérési értékét és ennek figyelembevételével adminisztrál és kezeli a képernyőt.

Az XSELECT kezelése elsősorban a DEMO.C program példáján keresztül érthető meg, az itt közölt leírás főleg referencia céljára szolgál.

Az XSELECT lehetőségei a következők:

Lehetőséget ad egy elem kiválasztására az UP, DOWN, PGUP, PGDN, HOME, END kurzorbillentyűk segítségével.

Ha az XSELECT második paramétere nem 0 (WINDSAVE-val lementett ablakban történik a tömbkezelés), lehetőséget ad

az ablak mozgatására a CTR_LEFT, CTR_RIGHT, CTR_PGUP és a CTR_PGDN kontroll-kurzorbillentyűk segítségével.

ESC leütése esetén -1 visszatérési értékkel adja vissza a vezérlést.

ENTER leütése esetén

- ha a 'sel_fp' függvénymutató 0 értékű, akkor visszatér a kiválasztott tétel indexével,
- ha a 'sel_fp' függvénymutató nem 0, akkor végrehajtja a mutatóval azonosított függvényt a következő paraméterekkel:
 - az aktuális tétel indexe
 - az aktuális sor a képernyőn.

A DEL billentyű hatására törli az aktuális tételt a képernyőről, ha a 'del_fp' függvénymutató nem 0, és végrehajtása után a visszatérési értéke 1.

Az INS billentyű hatására az aktuális tétel elé beszúr egy új tételt, ha az 'ins_fp' függvénymutató nem 0, és visszatérési értéke 1.

A CTR_INS vagy a CTR_A billentyű hatására az utolsó tétel után beszúr egy új tételt, ha az 'ins_fp' függvénymutató nem 0, és visszatérési értéke 1.

Az F1, F2, ..., F10 billentyűk hatására a funkcióbillentyűhöz rendelt függvényt hajtja végre (ha az nem 0 értékű).

Az XSELECT vezérlése egy struktúra definiálásával végezhető, amelynek elemei a következők:

char ul_row;	a használt képernyőrész (bal) felső sora
char ul_col;	a használt képernyőrész bal (felső) oszlopa
char lr_row;	a használt képernyőrész (jobb) alsó sora
char lr_col;	a használt képernyőrész jobb (alsó) oszlopa
int lastit;	az utolsó tétel indexe (0-tól számolva)
int def_it;	az alapértelmezett tétel indexe
XFP *fkeys;	a funkcióbillentyűkhöz rendelt függvények

XFP ini_screen; a képernyő-inicializáló függvény
 XFP to_screen; az egy tételt kiíró függvény
 XFP mark_it; a kiemelő függvény
 XFP sel_fp; a kiválasztott tételt 'módosító' függvény
 XFP left_fp; a balra nyíl hatására végrehajtandó függvény
 XFP rght_fp; a jobbra nyíl hatására végrehajtandó függvény
 XFP ins_fp; az egy tételt beszűrő függvény
 XFP del_fp; az egy tétel törlő függvény
 char col_base; az alapszín

Az egyes struktúraelemek jelentése:

ul_row (char)

az XSELECT által kezelendő képernyőrész felső sora.

ul_col (char)

az XSELECT által kezelendő képernyőrész bal oldali oszlopa.

lr_row (char)

az XSELECT által kezelendő képernyőrész alsó sora.

lr_col (char)

az XSELECT által kezelendő képernyőrész jobb oldali oszlopa.

lastit (int)

a tömb utolsó elemének indexe (ha a tömb induláskor üres, akkor -1).

def_it (int)

az alapértelmezés szerinti elem indexe az XSELECT indulásakor.

fkeys (XFP* or XFP[10])

a funkcióbillentyűkhöz rendelt függvények mutatóinak tömbje.

Ha az 'fkey' értéke 0, akkor az XSELECT a funkcióbillentyűkre nem reagál.

Az XSELECT csak arra a funkcióbillentyűre reagál, amelyhez a hozzárendelt függvénymutató nem 0.

Egy hozzárendelt függvény végrehajtásakor a következő két paramétert kapja:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

Ha a végrehajtott funkcióbillentyű függvényének visszatérési értéke nem 0, akkor az XSELECT végrehajtja az 'ini_screen' által mutatott függvényt, és újraindítja a képernyőt.
A funkcióbillentyűkhöz köthetők a nem az XSELECT hatáskörébe tartozó funkciók, például az 'fkeys' mutatótömb első elemébe (F1) a programozó a saját help rutinjának címét teheti (így a felhasználó számára a help az F1 billentyű leütésével érhető el).

ini_screen (XFP)

a képernyőt inicializáló függvény mutatója (XSELECT indulásakor hívódik).

A függvény paramétere:

a képernyőn az első tétel indexe.

to_screen (XFP)

a tömb egy elemét a képernyőre kiíró függvény mutatója.

Paraméterei:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

mark_it (XFP)

annak a függvénynek a mutatója, amely egy tételt kiemeltté (aktuális tételként megjelöltté) vagy nemkiemeltté tesz.

Az XSELECT-től kapott paraméterei:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn,

ha 'kiemelni' kell akkor 1, egyébként 0.

sel_fp (XFP)

a módosító függvény.

Ha ez a függvénymutató 0, akkor ENTER leütése esetén az XSELECT visszatér az aktuális tétel indexével.

Ha a függvénymutató nem 0, akkor a ENTER leütése esetén a függvény végrehajtódik a következő két paraméterrel:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

Ha a végrehajtott függvény visszatérési értéke nem 0, akkor az XSELECT végrehajtja az 'ini_screen' által mutatott függvényt, és újraindítja a képernyőt.

left_fp (XFP)

a balra nyíl leütése esetén végrehajtandó függvény.

Ha a left_fp függvénymutató nem 0, akkor az XSELECT a LEFT (balra nyíl) billentyű hatására végrehajtja a függvényt a következő paraméterekkel:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

Ha a függvény visszatérési értéke nem 0, akkor az aktuális tételre újra végrehajtódik a 'mark_it' függvény.

right_fp (XFP)

a jobbra nyíl leütése esetén végrehajtandó függvény.

Ha a right_fp függvénymutató nem 0, akkor az XSELECT a RIGHT (jobbra nyíl) billentyű hatására végrehajtja a függvényt a következő paraméterekkel:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

Ha a függvény visszatérési értéke nem 0, akkor az aktuális tételre újra végrehajtódik a 'mark_it' függvény.

ins_fp (XFP)

a beszúró függvény mutatója.

ha ez nem 0, akkor az XSELECT az INS és CTR_INS vagy CTR_A billentyűkre reagál.

INS esetén egy üres sort képez az aktuális sor előtt.

CTR_INS vagy CTR_A esetén egy üres sort képez az utolsó tétel után.

A függvény két paramétert kap az XSELECT-től:

az aktuális tétel sorszáma,

az aktuális sor a képernyőn.

Ha a beszúró függvény visszatérési értéke 0, akkor visszaáll a képernyő előző állapota.

Ha a függvény visszatérési értéke nem 0, a 'last_it' értéke eggyel nő, és az új tételre meghívódik a 'to_screen' függvény.

del_fp (XFP)

a törlési függvény mutatója.

Ha ez nem 0, akkor az XSELECT a DEL billentyű hatására meghívja

ezt a függvényt a következő paraméterekkel:

az aktuális tétel sorszáma,
az aktuális sor a képernyőn.

Ha a törlő függvény visszatérési értéke nem 0, akkor a 'last_it' értéke eggyel csökken, és az aktuális tétel törlődik a képernyőről.

col_base (char)
az alapszín.

Visszatérési érték

-2 hiba esetén,
-1 ESC leütése esetén,
ha a 'sel_fp' értéke 0, akkor ENTER esetén az aktuális tétel indexe.

Lásd még

select(), selfile(), <xselect.h>

Példa

Az XSELECT használatára a demonstrációs program főprogramja a példa.

4. Eltérések az MSCEX és ACEX környezetek között

A Microsoft C és az Aztec C nyelvekhez felépített könyvtárak között nincs lényeges eltérés.

A meglévő különbségek a következők:

- Az MSCEX könyvtár ablakkezelő függvényei az ablakhoz rendelt puffert a felhasználói adatszegmensen kívül, az ACEX megfelelői rutinjai a felhasználói adatszegmensen belül kezelik. E különbség miatt (Aztec C esetén), ha a lefordítandó C modulban szerepel a `<window.h>` header file beiktatása (`include`-ja), az Aztec compilert a `-dAZTEC` opcióval kell indítani (a csak Aztec-et használók értelemszerűen módosíthatják a `<window.h>` file-t).
- A header file-ok tartalmazzák a `LINT_ARGS` definíciókat, ezáltal az MSCEX igazodik a Microsoft C környezet szintax-ellenőrzési lehetőségeihez.
- Az ACEX könyvtár tartalmazza Microsoft C szabványos könyvtárában meglévő, de az Aztec C szabványos könyvtárából hiányzó stringkezelő rutinok egy részét:

```
strdup  
strupr  
strlwr
```

- A két nyelv közötti átvihetőséget az ACEX-hez tartozó `AZTEC.H` header file segíti egy makró halmazzal. Ennek használatakor az Aztec C compilert a `-dAZTEC` opcióval kell indítani.

5. A demonstrációs programok forráslistája

A demonstrációs programok Microsoft C környezetben reprodukálhatók. (Aztec C esetén a szabvány header file-ok cseréje szükséges, be kell illeszteni az <aztec.h> header file-t, azonkívül a compilert a -dAZTEC opcióval kell indítani.)

A négy program újrafordításának parancsai:

```
>MSC demo;  
>LINK /NOI demo,,,mscext  
>MSC dfiledir;  
>LINK /NOI dfiledir,,,mscext  
>MSC dbackmus;  
>LINK /NOI dbackmus,,,mscext  
>MSC dkeyscr;  
>LINK /NOI dkeyscr,,,mscext  
>DEL d*.map  
>DEL d*.obj
```

A DEMO program egy az XSELECT függvényen alapuló monitor, amely elindítja a másik három demonstrációs programot. A DEMO által bemutatott CEX függvények:

```
xselect  
curstype  
windsave,windclose,windfram,windmove  
cupo,erase,eol,eop  
aputtys,aputtysn,awrchar,puttys  
printhelp  
movframe  
setattr  
kbflush,kbget  
editline  
sound,nosound,delay
```

A DEMO indítás után az F2 billentyű hatására kiírja a kezelési lehetőségeket.

A DFILEDIR bemutatja, hogy miként kell megírni egy hierarchikus file-rendszert kezelő ablakos file-kiválasztást. A DFILEDIR által bemutatott CEX függvények:

```
currdisk
setcpos, aputtys, puttys
parsname
frstfile, nextfile
select
```

A DBACKMUS a háttérzenére ad példát. A bemutatott függvények:

```
windsave, windfram, windclose
cupo, aputtys
inimusic, endmusic
delay
kbstat, kbget
movframe
```

A DKEYSCR a billentyűzet és a képernyő kezelését mutatja be. A Programban hivatkozott CEX függvények:

```
curstype
cupo, setcpos
kbstat, kbshifts, kbget
setattr
erase
aputty, aputtys, aputtysn
rdchar
movframe
```

5.1. DEMO.C

```
/* DEMO.C
 *
 * monitor of CEX and demonstration of XSELECT
 *
```



```
* Robert Schill   Jul/28/1987   Produced by MI Software
*/
```

```
#include <string.h>
#include <malloc.h>
```

```
#include <window.h>
#include <setattr.h>
#include <xselect.h>
#include <editline.h>
#include <frame.h>
```

```
#define On      1
#define Off    0
#define NULL   0
#define YES    1
#define NO     0
```

```
#define DEFEXECNUM 3
```

```
/* A vegrehajtando programok (default) */
```

```
char *defexecs[DEFEXECNUM] =
{
    "DFILEDIR.EXE",
    "DBACKMUS.EXE",
    "DKEYSCR.EXE"
};
```

```
char *helps[] = /* a fentiekhez tartozo help szovegek */
{
    " DFILEDIR\r\n\
\r\n\
File es konyvtar kivalasztasat\r\n\
demonstralo program.\r\n\
\r\n\
A demonstralt rutinok:\r\n\
    frstfile,nextfile\r\n\
    parsname\r\n\
    puttys,aputtys,setattr\r\n\
    setcpo,cupo\r\n\
    select",
```



```

" DBACKMUS\r\n\
\r\n\
A hatterzenet demonstralo program.\r\n\
\r\n\
A demonstralt rutinok:\r\n\
    inimusic,endmusic\r\n\
    windsave,windfram,windclose\r\n\
    movframe\r\n\
    delay",

```

```

" DKEYSCR\r\n\
\r\n\
A billentyuzet es kepernyo\r\n\
kezeleset demonstralo program.\r\n\
\r\n\
A demonstralt rutinok:\r\n\
    curstype\r\n\
    cupo\r\n\
    aputtys\r\n\
    kbget,kbstat,kbshifts\r\n\
    rdchar\r\n\
    aputty\r\n\
    setattr"
};

```

/* A vegrehajtando programok nevei */

```

char **execs = NULL;      /* A vegrehajtando programok nevei      */
int execnum = 0;          /* A vegrehajtando programok szama      */

struct XSELECT rec;       /* XSELECT vezerlo      */
int chbase = COLOR(GRAY,BLACK); /* alapszin      */
int chighl= COLOR(RED,GRAY);  /* kiemelt szin      */
int cedit = COLOR(BLACK,RED); /* editalasi szin      */
int chelp = COLOR(BLACK,GREEN); /* help szin      */

int savecurs;             /* lementett kurzortipus*/
/* a kezelendo belso kepernyoterulet koordinatai */

#define ULR      8          /* (bal) felso sor      */
#define ULC      19         /* bal (felso) oszlop      */

```



```

#define LRR      16                /* (jobb) also sor      */
#define LRC      58                /* jobb (also) oszlop   */

#define MAXLEN   (LRC-ULC+1)      /* a mezo szelessege    */

/* az xselect altal hívható rutinok */

int clear();
int toscr();
int mark();
int exec();
int ins();
int del();
int help();
int info();

main()
{
    WINDTYPE iniscreen();
    WINDTYPE wsave;                /* a lementett ablak    */
    register struct XSELECT *prec = &rec; /* gyors eléréshez     */
    XFP f[10];                     /* funkcióbillentyűk rutinjai */

    wsave = iniscreen();
    savecurs = curstype(9,-1);     /* kurzor nem kell (EGA-ra is) */
    memset((char *)f,0,10*sizeof(XFP)); /* a többi 0 lesz!      */
    f[0] = help;                   /* F1                    */
    f[1] = info;                   /* F2                    */

    rec.ul_row    = ULR;
    rec.ul_col    = ULC;
    rec.lr_row    = LRR;
    rec.lr_col    = LRC;
    rec.lastit    = DEFEXECNUM-1;
    rec.def_it    = 0;
    rec.fkeys     = f;
    rec.ini_screen = clear;
    rec.to_screen  = toscr;
    rec.mark_it    = mark;
    rec.sel_fp     = exec;
    rec.ins_fp     = ins;

```



```

rec.del_fp      = del;
rec.col_base    = cbase;
rec.left_fp     = NULL;
rec.right_fp    = NULL;

iniexecs();                      /* a dinamikus tomb allokalasa */
xselect(prec,wsave);             /* vegrehajtas */
windclose(wsave);               /* a kepernyo visszaallitas */
curstype(savecurs >> 8, savecurs & 0xFF);
exit(0);
}

WINDTYPE iniscreen()
/* kepernyo inicializalas */
{
WINDTYPE wsave;                 /* a lementett ablak */

erase();
startproc();
wsave = windsave(ULR-5,ULC-1,LRR+1,LRC+1);
windfram(wsave,HDVS,cbase);
linsimple(ULR-1,ULC,LRC-ULC+1,cbase);
cupo(ULR-4,ULC);
aputtys(" DEMO: a CEX 2.2 Verzio demonstracioja",cbase);
cupo(ULR-3,ULC);
aputtys("      Schill Robert  (MI Software)",cbase);
cupo(ULR-2,ULC);
aputtys("      F1: help, F2: altalanos leiras",cbase);
return(wsave);
}

int help(act,lin)
/* F1 hatasara kiirja az aktualis info-t. (Az XSELECT hivja) */
int act,lin;
{
if (act >= DEFEXECNUM)
    return(NO);
printhelp(helps[act],chelp);
return(NO);
}

char infotxt1[] = "\

```



```

clear()
/* Torli az XSELECT belso kepernyoreszet, (Az XSELECT hivja) */
{
movframe(0,rec.ul_row,rec.ul_col,rec.lr_row,rec.lr_col,cbase);
}

```

```

int toscr(act,lin)
/* Kiirja az aktualis file-nevet, (Az XSELECT hivja) */
int act;

```

```

int lin;
{
register char *p;
register int len;
int col;

```

```

if (act >= execnum)
    return;
p = *(execs+act);
if ( (len = strlen(p)) > MAXLEN)
    len = MAXLEN;

```

```

col = ((rec.lr_col+rec.ul_col-len+1) >> 1)-1;
cupo(lin,col);
aputtysn(p,cbase,len);
cupo(lin,rec.ul_col);
}

```

```

int mark(act,lin,onoff)
/* Kiemeli az aktualis tetelt, (Az XSELECT hivja) */
int act;
int lin;
int onoff;
{
onoff = onoff ? chighl : cbase;
cupo(lin,rec.ul_col);
setattr(MAXLEN,onoff);
}

```

```

exec(act,lin)
/* ENTER hatasara vegrehajtja az aktualis programot.
(Az XSELECT hivja) */

```



```

int act, lin;
{
WINDTYPE wsave;
char *p;

wsave = windsave(0,0,24,79);
erase();
puttys("Vegrehajtasra kerul: ");
puttys(p = *(execs+act));
puttys("\r\n");
delay(3000);
system(p);
setcpos(0x1800);
eol();
puttys("Usson le egy billentyut...");
kbflush();
kbget();
windclose(wsave);
return(0);
}

```

```

ins(act, lin)
/* INS vagy CTR_INS hatasara uj file-nevet szur be.
(Az XSELECT hivja) */
register int act;
int lin;
{
register char *p;

cupo(lin, rec.ul_col);
p=editline(NULL, rec.lr_col, MAXLEN, cedit, chelp, '\r');
if (p == EL_NUL || p == EL_ESC)
    return(NO);

awrchar(MAXLEN, ' ', cbase); /* sor torlese */
execs = (char **) realloc(execs, (execnum+1)* sizeof(char*));
if (execs == NULL)
    fewmem();
if (act < execnum)
    memcpy( (char*) (execs+act+1),
            (char*) (execs+act ),
            (execnum-act) * sizeof(char *));

```



```

if ((p = strdup(strupr(p))) == NULL)
    fewmem();
*(execs+act) = p;
execnum++;
return(YES);
}

del(act,lin)
/* DEL hatasara az aktualis file-nevet torli.
(Az XSELECT hivja) */
register int act,lin;
{
if (rec.lastit <= 0)
    return(NO);

free(*(execs+act));
if (act < --execnum)
    memcpy( (char*) (execs+act),
            (char*) (execs+act+1),
            (execnum-act) * sizeof(char *));
execs = (char **) realloc(execs,execnum*sizeof(char **));
return(YES);
}

iniexecs()
/* Feltolti a DINAMIKUS stringtombot */
{
register char **ps = defexecs;
register char **pt = defexecs;
register int i = DEFEXECNUM;

if ((execs = (char **) malloc(i * sizeof(char*))) == NULL)
    fewmem();

for (pt = execs; i--; pt++)
{
    if ( (*pt = strdup(*ps++)) == NULL)
        fewmem();
}
execnum = DEFEXECNUM;
}

```



```

fewmem()
/* Keves memoria eseten hibauzenettel leallitja a programot */
{
setcpos(0x1700);
eop();
puttys("Keves a memoria!\7");
curstype(savecurs >> 8, savecurs & 0xFF);
exit(-1);
}

#define MSGWIND COLOR(0,(318))

startproc()
/* Mozgatással kiírja a MAGISZTER Szoftver feliratot */
{
WINDTYPE savewind; /* az ablak buffere */
int ur,lc; /* az ablak bal felső sora - oszlopa */
int lr,rc; /* az ablak jobb alsó sora - oszlopa */

/* a kezdőpozíció */
ur=1; lc=0; rc=20; lr=5;

if( (savewind=winsave(ur,lc,lr,rc)) == NULL) /* nem sikerült */
{
erase();
puttys("Keves a memoria!");
exit(0);
}

windfram(savewind,HSVS,MSGWIND); /* az ablak kerete */
cupo(ur+1,lc+3);
aputtys("M A G I S T E R",MSGWIND); /* attribútumos kiírás */
cupo(ur+3,lc+6);
aputtys("SOFTWARE",MSGWIND);

while(windmove(savewind,WM_RIGHT) == 0) /* az ablak jobbra */
;
while(windmove(savewind,WM_DOWN) == 0) /* az ablak lefele */
;
while(windmove(savewind,WM_LEFT) == 0) /* az ablak balra */
;
while(windmove(savewind,WM_UP) == 0) /* az ablak felfele */
;

```



```

sound(1200);delay(800);nosound();      /* hangjelzes      */
}

```

```

/* end */

```

5.2. DFILEDIR.C

```

/* DFILEDIR.C
 *
 * demonstration of file and directory selection (called by DEMO)
 *
 * Robert Schill   Jul/28/1987   Produced by MI Software
 */

```

```

#include <frstfile.h>
#include <parsname.h>
#include <select.h>
#include <setattr.h>
#include <string.h>
#include <malloc.h>

```

```

typedef struct FREC
{
    char *fn;
    char isdir;
} SFREC, *PFREC;

```

```

PFREC actfiles;      /* file-ok es könyvtarak faja */
int actdisk;          /* aktualis lemezmeghajto      */
char inspec[80];      /* input file-specifikacio     */
char fullspec[80];    /* teljes file-specifikacio     */
int cbase = COLOR(BLACK,GRAY); /* alapszin                    */
int chigh = COLOR(BLACK,RED);  /* kiemelt szín                 */
char *myrealloc();     /* ellenőrzött realloc         */

```

```

main(argc,argv)
/* Bemutatja a FRSTFILE,SELECT es egyeb függvények hasznalatát */
int argc;
char **argv;
{

```



```

int res;

actdisk = currdisk();

if (argc < 2)
    strcpy(inspec, ".*");
else
    strcpy(inspec, argv[1]);

while ((res = dodir(inspec)) > 0)
    strcpy(inspec, fullspec);
setcpos(0x1700);
if (res == -2)
    exit(-1);

if (res == -1)
    puttys("Nincs kiválasztott file");
else
    {
        puttys("A kiválasztott file: ");
        puttys(strupr(fullspec));
    }

exit(0);
}

dodir(spec)
/* Egy könyvtarat feldolgozo rutin */
char *spec;
{
    int toscr();
    int compare();
    register PFREC files = 0;
    PFREC oldfiles;
    int fnum = 0;
    struct FC fc;
    register struct FC *pfc;
    char *fname;
    int res;

    /* a kepernyore kiirto fv. */
    /* ket nev hasonlitasa (qsort) */
    /* az aktualis file-fa */
    /* free-hez */
    /* a file-ok szama */
    /* Disk Transfer Area */
    /* mutat'o a --ra */
    /* a file-nev eleje */
    /* a kiválasztott file indexe */

    if ((fname = parsname(actdisk, spec, fullspec)) == 0)
    {
        setcpos(0x1700);
    }

```



```

        putty("Hibas specifikacio: ");
        putty(spec);
        return(-2);
    }
    pfc = &fc;

/* eloszor a konyvtarak */

strcpy(fname,"*.");
if (frstfile(fullspec,pfc,SUBDIR))
    {
        do
            if((pfc->f_attr&SUBDIR) && strcmp(pfc->f_nam,"."))
                {
                    files = (PFREC)myrealloc(files,
                        (fnum+1)*sizeof(SFREC));
                    (files+fnum)->isdir = 1;
                    (files+fnum)->fn = strdup(pfc->f_nam);
                    ++fnum;
                }
        }
        while (nextfile(pfc));
    }

/* a file-ok beolvasasa */

fname = parsname(actdisk,spec,fullspec);
if (frstfile(fullspec,pfc,ARCHIVE))
    {
        do
            {
                if ((pfc->f_attr & SUBDIR) == 0)
                    {
                        files = (PFREC)myrealloc(files,
                            (fnum+1)*sizeof(SFREC));
                        (files+fnum)->isdir = 0;
                        (files+fnum)->fn =
                            strlwr(strdup(pfc->f_nam));
                        ++fnum;
                    }
            }
        }
    }

```



```

        while (nextfile(pfc));
    }
    if (fnum == 0)
    {
        puttys("Nincs ilyen file: ");
        puttys(fullspec);
        return(-1);
    }

/* rendezes: a könyvtarak nagybetusek, ezek lesznek az elejen */
qsort((char *)files, fnum, sizeof(SFREC), compare);

actfiles = files;
if ((res = select(fnum, 14, toscr, fullspec, cbase, chigh, 0, 0)) < 0)
{
    fullspec[0] = '\0';
    return(res);
}
if ((files+res)->isdir)                /* könyvtár a kiválasztott? */
{
    if (strcmp((files+res)->fn, "..") == 0) /* apa-könyvtár? */
    {
        strcpy(inspec, fname);
        fname = strchr(fullspec, '\\');
        *fname = '\0';
        fname = strchr(fullspec, '\\')+1;
        strcpy(fname, inspec);
    }
    else
    {
        strcpy(inspec, fname);
        strcpy(fname, (files+res)->fn);
        strcat(fname, "\\");
        strcat(fname, inspec);
    }
    res = 1;                          /* könyvtár -> folytatás */
}
else
{
    strcpy(fname, (files+res)->fn);

```



```

        res = 0;                                /* file kiválasztva */
    }
    for (oldfiles = files; fnum--; files++) /* a fájl felszabadítása */
        free(files->fn);
    free(oldfiles);
    return(res);
}

compare(f,s)
/* Ket fájl (vagy könyvtárneve) hasonlítása a QSORT számára */
PFREC f,s;
{
    return(strcmp(f->fn,s->fn));
}

toscr(act,lin,col)
/* egy fájl (vagy könyvtár) nevének kiírása a SELECT számára */
int act,lin,col;
{
    cupo(lin,col+1);
    if ((actfiles+act)->isdir)
    {
        aputtys("<",cbase);
        aputtys((actfiles+act)->fn,cbase);
        aputtys(">",cbase);
    }
    else
        aputtys((actfiles+act)->fn,cbase);
}

char *myrealloc(what,size)
/* ellenőrzött realloc */
char *what;
int size;
{
    char *realloc();
    char *p;

    if (what == 0)                                /* első allokálás? */
        p = malloc(size);
    else

```



```

        p = realloc(what,size);
if (p == 0)
{
    setcpos(0x1700);
    putty("Out of memory!\7");
    exit(-1);
}
return(p);
}

/* end */

```

5.3. DBACKMUS.C

```

/* DBACKMUS.C
 *
 * demonstration of back music
 *
 * Robert Schill   Jul/28/1987   Produced by MI Software
 */

#include <music.h>
#include <window.h>
#include <setattr.h>
#include <frame.h>

char boci[] = {
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    dd(3),mi(3),dd(3),mi(3),so(2),so(2),
    Dd(3),ti(3),la(3),so(3),fa(2),la(2),
    so(3),fa(3),mi(3),re(3),dd(2),dd(2),
    pa(2),
    fine};

#define SZOVEGHOSSZ 15
char szoveg[] = "Boci-boci tarka ";
char *szovegveg = szoveg+SZOVEGHOSSZ;

main()

```



```

/* A hatterzeneles demonstracioja */
{
WINDTYPE sw; /* lementett ablak */
register int i,c;
int savech; /* a szoveg-swap-hoz */

sw = windsave(8,30,14,47); /* ablaknyitas */
windfram(sw,HDVD,BLUE); /* keretezes */

for (i = 9,c = BLUE; i < 14; i++, c++) /* kezdó kiiras */
{
    cupo(i,31);
    aputtys(szoveg,c);
    savech = szoveg[0];
    memcpy(szoveg,szoveg+1,SZOVEGHOSSZ);
    *szovegveg = savech;
}

inimusic(boci); /* hatterzene inditasa */
while (!kbstat())
{
    movframe(1,9,31,13,46,GRAY);
    cupo(13,31);
    aputtys(szoveg,c);
    savech = szoveg[0];
    memcpy(szoveg,szoveg+1,SZOVEGHOSSZ);
    *szovegveg = savech;
    if (++c > WHITE)
        c = BLUE;
    delay(800); /* varakozas 8/10 s-ig */
}

kbget(); /* billentyuzetpuffer torlese */
windclose(sw); /* ablak zarasa */
endmusic(); /* hatterzene leallitasa */
}

/* end */

```


5.4. DKEYSCR.C

```
/* DKEYSCR.C
 *
 * demonstration of keyboard and screen handling
 *
 * Robert Schill    Jul/28/1987    Produced by MI Software
 */

#include <setattr.h>
#include <kbget.h>
#include <kbshifts.h>

char helptext[] = "\
Hasznalja a kurzorbillentyuket a mozgatashoz\r\n\
(es szinvaltashoz),\r\n\
es\r\n\
probalja ki a shift (lock) billentyuket!"

main()
/* A billentyuzet es kepernyokezeles demonstracioja */
{
int savecurs;           /* az elementett kurzortipus      */
register int c;         /* a leutott billentyu kodjai   */
int lin = 0;           /* aktualis sor                  */
int col = 0;           /* aktualis oszlop               */
int attr;              /* aktualis szin                 */
int stat;              /* shift statusz                 */
int oldstat = -1;      /* elozi shift statusz           */

savecurs = curstype(0,7); /* "nagy" kurzor                */
fill();
cupo(lin,col);

again:

while (!kbstat())       /* amig a billentyure varakozunk*/
{
    if ((stat = kbshifts()) != oldstat)
    {
```



```

        printstat(stat);
        oldstat = stat;
        cupo(lin,col);
    }

}

c = kbget();
switch (c & 0xFF)      /* also byte: ASCII kod      */
{
    case 0:
        switch (c >> 8)      /* felso byte: scan kod */
        {
            case UP:
                if (--lin < 0)
                    lin = 24;
                break;
            case DOWN:
                if (++lin > 24)
                    lin = 0;
                break;
            case LEFT:
                if (--col < 0)
                    col = 79;
                break;
            case RIGHT:
                if (++col > 79)
                    col = 0;
                break;
            case HOME:
                if (lin == 0 && col == 0)
                    goto again;
                lin = col = 0;
                break;
            case END:
                if (lin == 24 && col == 0)
                    goto again;
                lin = 24;
                col = 0;
                break;
            case PGUP:
                if (lin == 0 && col == 79)

```



```

        goto again;
        lin = 0;
        col = 79;
        break;
    case PGDN:
        if (lin == 24 && col == 79)
            goto again;
        lin = 24;

        col = 79;
        break;
    case F1:
        printhelp(helptext,
                  COLOR(BLACK, GREEN));
        goto again;
    default:
        goto again;
}

cupo(lin, col);
attr = (rdchar() >> 8); /* szin beolvasasa */
if (++attr > COLOR(WHITE, GRAY))
    attr = BLUE;
setattr(1, attr);
goto again;

case ESC:
    erase();
    curstype(savecurs >> 8, savecurs & 0xFF);
    exit(0);

default:
    c &= 0xFF;
    if (c >= ' ')
    {
        aputty(c, attr); /* egyet lep is */
        if (++col > 79) /* de adminisztraljuk is */
            col = 0;
        cupo(lin, col);
        attr = (rdchar() >> 8);
    }
    goto again;
}
}

```



```

char *shstats[] =
{
    "Right Shift","Left Shift","Control","Alt",
    "Scroll lock","Num lock","Caps lock","Ins lock"
};

printstat(stat)
/* Kiirja a shiftallapotot */
register int stat;
{
    register int work = 1;
    int i;

    movframe(0,0,65,8,79,GRAY);          /* torles      */
    for (i = 0; i < 8; i++)
    {
        cupo(i,67);
        aputtysn(shstats[i],(stat&work)?COLOR(BLACK,RED):GRAY,11);
        work <<= 1;
    }
}

char buff[] = "\
0123456789012345678901234567890123456789\
0123456789012345678901234567890123456789";

fill()
/* Feltolti a kepernyot */
{
    register int i = 25;
    register char *p = buff;

    setcpos(0);                          /* HOME! */
    while (i--)
    {
        aputtys(p+1,GRAY);
        memcpy(p,p+1,80);
        *(p+80) = (i == 1) ? '\0' : *p;
    }
}

/* end */

```


AKADÉMIAI KIADÓ ÉS NYOMDA
Felelős kiadó: Hazai György főigazgató
Felelős szerkesztő: Czere Károlyné
A borítót Szabadi Zoltán tervezte
ISBN 963 05 4739 2
A kézirat lezárva: 1987. szeptember
OPTIMUM—MGKSZ NY. 87. 353.

CEX -- mint C EXTension,

vagyis a C programozási nyelv kiterjesztése, amely kényelmesebbé teszi a rendszerprogramok és fejlett alkalmazói programok írását.

A Közgazdasági Egyetem fiatal munkatársainak, Schill Róbertnek és Szegedi Jánosnak, CEX programkönyvtára az alábbi szolgáltatásokat nyújtja:

fejlett billentyű és képernyőkezelés
ablaktechnika
RS232 vonalkezelés
lemez és könyvtárkezelés

A CEX rendszer IMB/PC/XT/AT kompatibilis számítógépeken, DOS 2.00-től kezdődő operációs rendszerben, a Microsoft C 4.00 és AZTEC Manx C 3.20 programnyelvi környezetben működik.

A programlemez a könyvtárakon kívül az ismerkedést és betanulást segítő demonstrációs programot tartalmaz.

A CEX alkalmazásával már számos működő rendszer készült, sorozatunkban például a CFIO/SFIO bemutató programjai.

A könyv szövege a DOG dokumentáció generátor segítségével készült.

Fogy. ár: 8000 Ft.



