MINIX HOW TO instalation under WinXP

1)install Bochs 2.3.7 - full install wraz z dlx linux!
2) Zawartosc katalogu C:\Program Files\Bochs2.3.7\dlxlinux przekopiowac do C:\Program Files\Bochs2.3.7\minix2
3) używając np. WinSCP ściągnąć pliki (minix203.img,floppy.img) z folderu roboczego na galerze – galera.ii.pw.edu.pl (logując się jak na laborce). Nastepnie skopiowac je do naszego minix2.
4) edytujemy plik C:\Program Files\Bochs-2.3.7\minix2\ bochsrc.bxrc
Jego NOWA POSTAĆ:

```
##############################################################
# bochsrc.txt file for DLX Linux disk image.
##############################################################

# how much memory the emulated machine will have
megs: 128

# filename of ROM images
romimage: file=../BIOS-bochs-latest
vgaromimage: file=../VGABIOS-lgpl-latest

# what disk images will be used
floppya: 1_44=floppy.img, status=inserted    #to aby mozna uzywac dyskietki floppy.img
#floppyb: 1_44=floppyb.img, status=inserted

# hard disk
ata0: enabled=1, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14
ata0-master: type=disk, path="minix203.img", cylinders=615, heads=6, spt=17
#ata0-slave: type=cdrom, path=minix2.iso, status=inserted   #jesli boot z .iso


# choose the boot disk.
boot: disk
#boot: cdrom    #jesli boot z .iso


# default config interface is textconfig.
#config_interface: textconfig
#config_interface: wx

#display_library: x
# other choices: win32 sdl wx carbon amigaos beos macintosh nogui rfb term svga

# where do we send log messages?
log: bochsout.txt

# disable the mouse, since DLX is text only
mouse: enabled=0

# enable key mapping, using US layout as default.
```

```
#
# NOTE: In Bochs 1.4, keyboard mapping is only 100% implemented on X windows.
# However, the key mapping tables are used in the paste function, so
# in the DLX Linux example I'm enabling keyboard_mapping so that paste
# will work.  Cut&Paste is currently implemented on win32 and X windows only.

#keyboard_mapping: enabled=1, map=$BXSHARE/keymaps/x11-pc-us.map
#keyboard_mapping: enabled=1, map=$BXSHARE/keymaps/x11-pc-fr.map
#keyboard_mapping: enabled=1, map=$BXSHARE/keymaps/x11-pc-de.map
#keyboard_mapping: enabled=1, map=$BXSHARE/keymaps/x11-pc-es.map
```

najwazniejsze sa te miejsca pogrubione. #to komentarz

5) edytujemy plik C:\Program Files\Bochs-2.3.7\minix2\ run.bat
Jego nowa postac ma byc:

```
cd "C:\Program Files\Bochs-2.3.7\minix2"
..\bochs -q -f bochsrc.bxrc
```

6)w tym momencie uruchamiamy emulator za pomaca run.bat. wyskakuje BLAD:
PANIC ……
ata0-0 disk size doesn't match specified geometry
-nie kill’ujemy procesu, robimy continue!
-jak się okazuje wszystko działa jak na laboratorium.

POMOC DO LAB1

//procedura obslugi syscallla (trzeba to wklepac w /usr/src/mm/main.c np. gdzies na koncu)

```
int do_getprocnr(void)
{
int procNum;
for(procNum = 0; procNum < NR_PROCS; ++procNum){
/*tu możecie dostac dodatkowe zadanie polegające na tym ze np. dla procesu o pid = 15
należy zwracac zawsze 999 –robimy to tak: odkomentuj ponizsza linijke:
if(pid==15) return 999;
*/
if( ( (mproc[procNum].mp_flags & IN_USE) != 0) && (mproc[procNum].mp_pid == pid) )
return procNum;
}
return ENOENT;
}
```

funkcja testujaca

-cd

–cd root

-vi main.c

//w tym momentcie w folderze root tworzymy(o ile nie istnieje) plik testujący nasza nowa funkje systemowa – można to zrobic rownie dobrze w innym folderze

kod w root/main.c

```
#include </usr/include/lib.h>
#include </usr/include/stdio.h>

//funkcja testujaca
int main (int argc, char* argv[])
{
  int result;
  int i;

  if(argc == 1) printf("Argument needed!\n");
  else{
     int value = atoi(argv[1]);
     for(i=0;i<10;i++)
          {
           result = getprocnr(value+i);
            if(result !=-1) printf("Indeks procesu o pid: %d to %d\n",value+i,result);
            else  printf("Procesu o pid: %d nie ma w tablicy! Kod bledu: %d\n",value+i,errno);
           }
    }
 return 0;
}
```

```
//funkcja uzytkownika do wywolywania syscalla
int getprocnr(int proc_id)
{
message msg;

  msg.m1_i1 = proc_id;
  return (_syscall(0,78,&msg));  /*tu juz musi istniec nasza funkcja systemowa*/
}
```

dodatkowe zadanie do wykonania aby całość zadania zadziałała:

-w callnr.h na poczatku zmienic 78 na 79
na koncu dopisac #define GETPROCNR   78
-w proto.h dodac na koncu linie:
_PROTOTYPE(int do_getprocnr, (void)   );
-w mm/table.c dodac po ostatniej komendzie przed klamerka }; :
do_getprocnr,   /*78 */
-w fs/table.c dodac tam kawalek za srodkiem ponizej "do_svrctl, /* */" dopisac linie(tuz przed };)
no_sys,

W RAZIE PROBLEMOW Z REKOMPILACJA: wykonac następujące kroki po starcie
- cd
- cd minix
- rm *
- cd
- cd usr/src/tools
- make clean
-make hdboot   //tutaj sie troche zejdzie
-cd
-shutdown
-boot
opcjonalnie: cd root i cc main.c
./a.out 33   -wywolac testowanie z parametrem

UZYWANIE DYSKIETKI

Dyskietka to plik o nazwie floppy.img który kopiujemy z serwera na laborce0 wraz z obrazem minixa minix203.img

uruchom normalnie minixa i wpisuj co nastepuje
mkfs /dev/fd0 1440 //to sie wpisuje tylko za pierwszym razem (to jest jakby format a:)
mount /dev/fd0 /mnt //montujemy
w katalogu mnt masz teraz to co będzie zapisane w wirtualnym obrazie floppy.img
czyli robimy dalej tak
cd root
cp main.c /mnt
cp a.out /mnt
i masz juz oba pliki które zrobiono wczesniej na laborce na floppy.img
na koniec koniecznie
cd
umount /dev/fd0
sync
<ctrl-D> //opcjonalnie wylogowanie
teraz w pliku floppy.img sa twoje 2 pliki stworzone na laborce
możesz je wysłać do kogokolwiek, który gdy zamontuje floppy.img (nie wykonac mkfs!! Bo utracimy nasze pliki, tylko mount itd…) to będzie miał w swoim folderze /mnt nasze pliki.