Initializing a Ceres/Oberon System

The only program residing in a "bare" machine is the boot loader in ROM. It is automatically initiated when power is switched on or when the reset button is pressed. It uses no peripheral devices except the one that is selected for input according to the middle mouse key on Ceres–1 or the external switch on Ceres–2.

0. The boot file is loaded from diskette.

1. The boot file is loaded from disk.

Four different boot files are available for different purposes (*x* stands for 1, 2, 3, or 3D):

0. *Ceresx.CheckBoot.* It is used to initialize the disk and to check and correct consistency of the file system.

1. *Ceresx.Boot0.* It requires that the file system is in a consistent state (possibly empty), and is used to load system files from diskette. They are typically available on the same diskette which contains *Ceres.Boot0.*

2. *Ceresx.Boot.* It requires a consistent state of the file system and a number of system files. It is the regular boot file and is typically loaded onto the boot track of the hard disk.

3. *Ceresx.ScavBoot.* It is used when the file directory has been corrupted and restores it.

Disk Initialization

In order to initialize a disk, insert the cassette labelled *Ceres.CheckBoot* and execute the following steps. The program accepts one–character commands, of which the "dangerous" ones are preceded by an exclamation mark (!). Parameters are names and numbers. When making an error in typing, a DEL causes the previous characters to be ignored; the entire parameter has to be repeated. ESC rescinds the command.

0. Type !F . This causes the disk to be formatted, a process which takes about 15 min. Termination is signalled with ">", and the program is ready for the next command.

1. Type !? . This causes the disk to be checked. The numbers of unusable (bad) sectors are listed. If they are already identified (given by the disk manufacturer), this step may be omitted. You may try to correct bad sectors (see Memo *Checking the disk using Ceres.CheckBoot*).

2. Type !1 . This causes the file directory to be initialized (at sector 1).

3. Type !2 . This initilizes the bad sector table (at sector 62).

4. Type b . Then enter the numbers of the bad sectors; each number is followed by a blank, the last by RETURN.

System Initialization

The disk is now in a consistent state and bad sectors have been set aside. Now the system files necessary to start Oberon must be loaded by executing the following steps:

0. Insert the diskette labelled Oberon1 (containing *Ceres.Boot0*) and press reset.

1. Type T . Enter date (form: 17:33:24 22.02.89)
2. Type z . Now all files on the diskette are loaded.
3. Type !B . The file *Ceresx.Boot* is copied onto the boot track of the disk.

This terminates system initialization. Setting the boot switch to select the disk and pressing Reset will start the Oberon system.

InitCeres.Text

Installing Bad Sector Table and Correcting Bad Sectors

Most disks contain a few sectors that are unusable (bad). In order to avoid their allocation by the Oberon System, they need to be set aside by collecting them in a file (called BadSectors). This file is present on all Ceres computers (except Ceres3/Ceres3D). If a bad sector is detected by malfunctioning, the following steps are needed to insert it in this file:

0. Insert diskette containing *Ceres.CheckBoot* and push Reset.

1. Type x (followed by the number of the bad sector). Saves the track containing the sector.

2. Type !Y. Formats the track.

3. Type !Z . Restores the track.

4. Type x (followed by the same number). If the displayed sequence of characters does not contain %, the bad sectors have been corrected. Otherwise repeat steps 1 – 3. If the %–sign persists, the track contains a "hard" error, and the sector has to be set aside by performing step 5.

5. Type b, then enter the numbers of the bad sectors that were uncorrectable, a blank after each number, RETURN after the last. (Mistyping can be corrected with DEL; thereafter the entire number has to be retyped.)

BadSectors.Text

Checking the Disk using DiskBoot

Introduction

The boot program *Ceresx.DiskBoot* may be used to check the consistency of the file system and to correct inconsistencies. (*x* stands for 1, 2, 3, or 3D). It must be used with utmost care.

The root of the Oberon file system is its directory, which is organized as a B–tree. Each page of the tree occupies one disk sector and contains the following information:

0. A mark to designate the sector as a directory page.
1. An integer m specifying the number of file names on the page. It lies between 12 and 24, except for the root page, where it is between 0 and 24.
2. m entries, each consisting of a file name, the address (sector no.) of the file's header sector, and m+1 addresses of descendant pages.

All descendant addresses on a page are either 0, in which case there are no descendants, or not 0. The root page is allocated on sector 1.

Each file, represented in the directory as a name/address pair, occupies a number of sectors, the first of which is the header. The header contains the following information:

0. A mark, identifying the sector as a file header.
1. The file name.
2. The password code of the user who created the file.
3. Time and date of creation (encoded).
4. The length of the file in bytes.
   aleng = no. of sectors used –1; *bleng* = no. of bytes on last sector + headersize.
   Total length = aleng * sectorsize + bleng – headersize
   (sectorsize = 1024, headersize = 352)
5. A sector table.
6. An extension sector table.

The addresses (sector no.) of the first 64 sectors are listed in the sector table. Further sectors are listed in extension sectors, each containing at most 256 entries. Each extension sector is listed in the extension sector table of the header.

The size of the extension table is 12, so the maximum number of data sectors is 12*256+64 = 3136, and the maximum length of a file is 3136*1024 – headersize = 3'210'912 bytes.

Commands

Commands are identified by a single character, except that the dangerous (writing) commands are prefixed with an exclamation mark (!). Parameters are names and numbers. A DEL causes the typed digits to be ignored; the entire parameter must be retyped. A number may be prefixed by a quote ('), in which case it is interpreted as hexadecimal. ESC rescinds the command.

?	List available commands.
l	Clear display.
r	Read a sector into sector buffer and display it as hexadecimal numbers.
R	Read a sector into sector buffer and display it as ASCII characters.
d	Read a sector into directory buffer and display it as a directory page.
h	Read a sector into header buffer and display it as a header.

e Read a sector into sector buffer and display it as an extension sector.

Changing a data sector after the r command:

i Insert number in sector buffer. The number is not written on disk by this command. If the value entered is followed by a blank, a next value is expected and entered into the next loaction; if it is followed by RETURN, the command is terminated.

!W Write the buffered sector onto disk.

Changing a directory sector after the d command:

n Insert name in directory buffer.

S Insert sector number in directory buffer.

!D Write directory buffer onto disk.

Changing a header after the h command:

s Insert sector number into sector table of header buffer.

L Insert length (aleng) in header buffer.

!H Write header buffer onto disk.

Fixing bad sectors:

x Read track into track buffer. A track is specified by the sector number of any of its sectors.

!Y Format track. This operation destroys all data on this track.

!Z Restore track from track buffer.

Miscellaneous commands:

f Find header of specified file.

c Check the consistency: list bad directory sectors and specify invalid sector numbers.

q Find all files containing specified sector.

b Add bad sectors to bad sector table (sector 62). A list may be entered, each number being followed by a blank. RETURN terminates the command.

!0 Clear sector.

!1 Initialize the directory. After this command, the directory contains the single file BadSectors.

!2 Initialize the bad sector table. The table is empty.

!? Check the entire disk for bad sectors. (Takes about 15 min.)

!F Format the entire disk. (All data are lost; takes about 15 min.)

DiskCheck.Text