PERQ File System Utilities Manual

Brad A. Myers

This manual describes the PERQ file system utilities: Partition, Scavenger, MakeBoot, and FixPart. Other utilities are described in the manual "PERQ Utility Programs Manual." In addition, relevant parts of the file system are described so the programs can be understood.

1. Preface: Notation Conventions.

The notations used below have been clearly and consistently used throughout this document.

SYMBOL	MEANING
[]	optional feature
{}	0 to n repetitions
CAPITALS	literal
lowercase or LowerCase	metaname
^	control key
CR	carriage return
SHIFT	shift key
¦	or

2. Introduction.

The PERQ has a hierarchical, multi-directory file system which supports search lists and noncontiguous files. Devices (e.g., hard and floppy disks) are divided into a number of sections called "partitions." Each partition can contain any number of directories. The main directory in a partition (the one you get if you simply specify the partition name) is the Root directory for that partition. All directories can contain other directories and files. Directories are stored as standard files and can be accessed by any program. However, the system knows special things about their format. All files in this file system can be noncontiguous; blocks for files may be scattered throughout the partition in which they reside. The naming conventions for filenames are described in detail in "The PERQ Introductory User Manual" but all are of the form:

```
    [[[device]:partition]>]{directory>}[filename]
```

Files may be stored on floppy disks in two ways. The FLOPPY program can be used to transfer files to and from the floppy, or the floppy can be initialized as part of the file system. These types of floppy disks are called "FLOPPY floppies" and "filesystem floppies" respectively.

NOTE: Floppies of one type CANNOT be used for the other; the formats are totally incompatible. Floppy disks that contain files transferred by the program FLOPPY must be accessed by the FLOPPY program (and not by the file system directly), and FLOPPY cannot read a file system floppy.

To create a file system floppy, use the Partition program described below.

The user programs that handle the file system are described in the "PERQ Utility Programs Manual" and only Partition, Scavenger, MakeBoot, and FixPart are described in detail here.

To create a file system on a blank disk, it is necessary to run the command file "InitDisk.Cmd" which is found on the boot floppy in every release package. A heavily-commented printed copy of this command file is provided as instructions for how to initialize a blank device. To create a file system on a floppy, use the Partition program described below. The manual "How To Make a New System" describes the process necessary to create a system using the MakeBoot program. This manual provides the background necessary to understand these processes.

3. Partitions and the Partition Program.

The partitions on a device are restricted to be fewer than 32768 blocks, where each block is 256 words (512 bytes, 4096 bits). The block size is fixed for all devices. On a 12-megabyte disk or a floppy, the entire device can be in one partition. On a 24-megabyte disk, at least 2 partitions are needed. We recommend, however, that each partition have 10080 or fewer blocks in them, otherwise Scavenger (described below) cannot handle the partition in one pass. All partitions must start and end on cylinder boundaries; the Partition program enforces this constraint.

No file or directory can be in more than one partition (a file cannot cross a partition boundary). Therefore, having free blocks in one partition does not prevent you from running out of free blocks in another.

The Partition program is used to create and modify the partitions on a device. Creating a partition usually destroys all old data in the area where the partition is made. The device should first be formatted (by using DTST on hard disks or FLOPPY for floppy disks). After formatting, Partition is the first program to run.

Partition asks whether you want to "debug". If you answer "YES", then nothing will be modified on the device. Answer "NO" if you want to make modifications.

Next, Partition requires that you tell it what device you want to modify. The choices are the harddisk or floppy. If you specify the harddisk, Partition checks to see whether the disk is a 12 or 24-megabyte disk. It then asks for confirmation of the size chosen. If you specified the floppy, you then have to tell partition whether the floppy is or single or double-sided. A floppy that is formatted on both sides can be partitioned as either single or double sided.

Next, the program asks if you want to partition the entire disk. Answer "YES" if you are starting from scratch (for example, on a newly formatted floppy or when installing the file system on a new machine). If you want to modify an existing device, answer "NO".

If you answered "YES" to initialize the entire disk, then the program asks for information about each partition in turn. The device must first be given a name (eight or fewer characters). Next, each of the partitions must be given a name (also eight or fewer characters). Examples of partition names used for the hard disk include "boot", "user", and "exp". There is no problem with having a partition with the same name as a device, but all the partitions must have unique names. The size of each partition is also requested. As mentioned earlier, we have found that 10080 or fewer blocks are desirable; otherwise Scavenger (described below) cannot handle the partition in one pass. This means there are 3 partitions on a 12-megabyte disk and 5 on a 24-megabyte disk. The minimum size for partitions is 120 blocks on a 12 megabyte disk, 240 blocks on a 24-megabyte disk, and 6 blocks on a floppy disk. The sizes of all partitions must be multiples of the mimimum size for that device.

For each partition on the device, the program asks whether to initialize the partition pages. This process puts all the pages in the partition on the free list and therefore should be done for a new device. The next question is whether to test after initializing the pages. Although this slows down the initialization somewhat, it is good practice to do this testing. If any pages are found to be bad during testing, they are removed from the free list so they will never be accessed again. The final question asked before initialization is whether to write every page twice. This provides some additional protection from bad pages since random data is written into the header and body of each block. It has been found in practice that some bad blocks on the disk will pass the first test and fail this one.

After all the blocks on the device have been included in a partition, the program displays some data about the device and asks whether to remount the device. Only mounted devices can be accessed, so if you plan to use the device, say "YES".

If you said that you did not want to initialize the entire device, the program reads the device information block (which is in a reserved, fixed location on cylinder 0) to find what partitions are currently on the device. These are displayed in the order they appear on the disk. The first question asked is whether to rename the device.

NOTE: Before renaming a device or partition it is important to note that every Run file on the device has incorporated in it the device and partition name where it was linked. If you rename the device, then no program on that device can be run (including the Shell, Login, Partition, for example). If you rename a partition, no program in that partition can be run.

If you did not rename the device, you are asked which partition you want to modify. Type the name of one of the partitions. The program then asks what you want to do to the partition. You can split a partition into two parts, merge a partition with another, initialize a partition, or change the name of a partition.

It is never a good idea to split a partition with files in it since files cannot cross partition boundaries. Any file which has blocks in both partitions will be destroyed. It is safe to split an empty partition or one that you plan to erase. The program asks how much of the partition to leave with the old name; the rest of the blocks in the partition will be put in the partition just created. The program will ask if you want to initialize the partition pages to create a new free list. This is recommended. Next, a name for the new partition is solicited and the pages are initialized if desired.

It is much safer to merge two partitions together than to split one apart. You specify the first partition and it is joined with the partition which is next on the disk. This is the only time the order of the partitions on the disk matters. If you want to erase the new, bigger partition, say that you want to initialize the pages. If not, then all the files on both partitions can be saved. Just after Partition exits, you must run Scavenger program (see below) on the new partition. When running the Scavenger in this case, be sure to tell it to rebuild the directories so that the directories of the two partitions can be joined together.

Partition also can change the name of a partition. Do not change the name of the partition that is used in the current path since the default path name then becomes invalid. In addition, all entries in the search list that refer to the partition renamed will no longer work. After a rename, therefore, the system may not be able to find the Shell or any other programs.

Finally, you can initialize the partition. This is a fast way to delete all the files in the partition. After asking whether to initialize the partition, the program asks whether to initialize the partition pages. If you answer "NO", you must use Scavenger program to recreate the directory immediately after running Partition. There are few reasons to initialize the partition without initializing the pages.

The Partition program can take a switch on the command line. If it is invoked with the /BUILD switch, then the entire device is partitioned. Note that the arguments must be specified on the command line. This is a dangerous thing to do and it is only recommended for command files which bring up an entire disk. The format for this switch is

```
        Partition/Build <dev> <deviceName> <Partname> <PartName> ...
```

where <dev> is either "H" for the hard disk or "F" for the floppy. The device name is given next followed by enough partition names for the entire device. The partition names are used in the order specified so the first name will be used for the first partition on the disk. All partitions except the last will be the standard size (10080 blocks). If extra partition names are specified, they are ignored. If the device seems to be already formatted, the program requires confirmation before erasing the device. If the user answers NO, then partition is run the normal way asking the user all the questions.

4. The Scavenger Program.

The Scavenger program fixes a partition on a device that contains useful files. It checks all files for consistency, rebuilds the free list, and creates a new directory structure for the partition. The Scavenger should only be run on devices that have already been initialized by the Partition program. The Scavenger checks and fixes some system information and then checks all files in a partition for consistency and recreates the free list. It can also recreate the directories. The Scavenger also removes bad boots. The Scavenger should be run whenever an inconsistency is found in the file system or when some program aborts and asks you to run the Scavenger.

WARNINGS:

1) As with the Partition program, do not type Control-C (^C) to Scavenger after it has begun writing on the device. During the read pass, it is safe to ^C. If you type ^C after it begins rebuilding the directory, you may not be able to access anything in the directory. If this happens, rerun Scavenger from another partition.

2) Scavenger will not be able to recreate the directory if there are no free blocks in the partition. Therefore, if your partition is full, you must delete some files before running Scavenger. If you cannot delete any files due to a bad directory and there are no free blocks, then there is currently no way to rebuild the directory. In this case, you must initialize the partition, thus losing all files there. Fortunately, we have never seen this happen in practice.

Scavenger program fixes one partition at a time. It is possible (and sometimes necessary) to scavenge partitions other that the one you are currently running in. It is usually safe, however, to scavenge the current partition.

Scavenger program has three separate phases. In phase one, it checks and updates some of the system information. In this phase, bad boots are deleted. If a boot has been defined by MakeBoot (see below) but either the microcode or system code files have been deleted, the boot is known to be bad. The Scavenger cannot tell, however, if a boot file is deleted and another created in the same place before Scavenger is run. In this case, the boot will seem valid but will not work.

During the second phase of the scavenge, the partition specified by the user is checked for consistency. All blocks are read and a new free list is generated in ascending disk order (the old free list is discarded). In addition, blocks that are not readable are marked as "bad", and if they cannot be rewritten, they are marked as "incorrigible" and removed from the file system. All blocks that were in files containing bad or incorrigible blocks are put in the bad file. In addition, any malformed chains are added to the bad file. The user is asked for a name for this bad file in phase three of Scavenger.

In phase three, the directories for the partition are completely rebuilt. Scavenger will delete the old directories, if desired. Otherwise, the old directories are marked as such and their names have a "$" added to the end. If there is not enough room for copies of all the directories in the partition to be created, Scavenger will crash leaving the directories only partially created. In this case, you have to delete some of the files in the directory and then rerun Scavenger.

Before entering any name in a directory or creating a new directory, Scavenger first checks to make sure the name seems reasonable as a filename. Certain characters are not allowed in filenames. These include any control characters, "<", "/", ":", and " ". In addition, the name may not end with a ">" or contain ">..>" or ">.>". If a bad name is found, or two files have the same name, Scavenger requests a new filename from the user. After Scavenger is finished, you can examine the files with bad names to see whether they contains any useful information. If so, rename or edit the files to recover the data. Otherwise, simply delete the files.

The Scavenger in this pass also makes sure the length of all files are correct and allow you to specify a new length. Note that this refers to the stored length rather than the number of blocks in a file. Certain files, like directories and swap files, do not bother to set the length field. File lengths usually become wrong when the file is opened and written but not closed properly. This, for example, happens when a transfer is aborted.

The Scavenger asks the user a number of questions before it begins processing the partition. First, it asks whether to look at the floppy or hard disk. It then checks that device to see how big it is and then asks if its choice is correct.

When it has this information, Scavenger asks if it can make changes to the device in the first two phases of the program (the directory fixing is handled later). This is like the "debug option" for the Partition program. If you answer "NO", then Scavenger checks the partition for errors and reports them but does not fix anything. If you are running Scavenger only to fix the directory, it is about twice as fast to answer "NO" to this question; otherwise, "YES" is a good idea.

Next, Scavenger asks if it should do logical block number consistency testing and serial number consistency testing. The header of every block contains information about the state of that block. The information includes the count of the block in the file (is it the first, second, etc.) and a two-word identifier for the file the block belongs to. These numbers are checked for correctness if you answer "YES". If the Scavenger continually aborts due to FullMemory, answer "NO" to one of these questions. To avoid the FullMemory condition on machines with 256k bytes of main memory, the default for serial number checking is no. Just type CR to get the default answer.

Then Scavenger asks if there is enough memory to do the scavenge in one pass. If your partition has 10080 or fewer blocks in it and if the screen has been shrunk (the Shell shrinks the screen when it knows you are running the Scavenger), then the answer is "YES". If your partition is bigger than 10080 blocks, then answer "NO". Three passes will then be used and the program will be correspondingly slower.

The next question is how many retries for a suspect read. The default is 15. If the Scavenger aborts with the error "block xx was found bad during... but was thought to be good before" or if it aborts during rebuilding of the directories, try rerunning the Scavenger with a smaller number. One is the smallest valid answer.

If you answered "YES" when asked if Scavenger could change your disk, you will be asked three more questions about ways Scavenger might change the disk. First, Scavenger asks whether it should delete temporary files. Temporary files exist for swapping; all user files are permanent. Second, the Scavenger asks if it should delete old bad segments. As described above, files with bad blocks in them are marked as bad. If you answer "YES" to this question, then the bad file created by the previous scavenge of this partition is added to the free list. Finally, Scavenger asks if it can rewrite bad blocks. If a block cannot be successfully read in the specified number of retries, it is possible that writing new data onto the block will fix the problem. However, for our hard disks, this seems to have a small chance of fixing the problem. Therefore, the default answer is "NO". However, if you answer "YES", the bad blocks will be rewritten. If the write or a subsequent read fails, then the block is "incorrigible", otherwise it is "bad". If rewritting is not permitted, then the block is marked "incorrigible" as soon as the read fails. If there are only transient read errors, the block is left alone.

After you answer all of the questions, Scavenger can get to work. The title line of the window is updated to show what Scavenger is working on. In addition, various cursors are used to show the progress of the different passes. First, Scavenger does phase one checking as described above, fixing any discrepancies if you allowed changes. If Scavenger finds a problem with the partition or device information blocks it cannot fix, it asks for help. If you cannot figure it out either, the problem may be that: 1) you specified the wrong device type to the first question; 2) the device is not a file system device (e.g. a FLOPPY floppy); 3) the device has not been initialized; or 4) the device has been messed up beyond repair. Unfortunately, the only fix in this case is to re-partition the device from scratch.

After the device and partition information checks out, Scavenger displays a list of the partition names and asks which one it should work on. Type the name of the partition to be scavenged. Next, Scavenger will display some information about the specified partition. The values are those stored in the partition information block before the scavenge. Now Scavenger makes a read pass through the partition building tables of each block's next and previous link (this is the data usually visible in the lower portion of the screen). The pass is done in eight parts for efficiency. Therefore, the Scavenger cursor goes down the screen eight times before the next step. The tables built by Scavenger are now checked for consistency, and the cursor changes to show that checking is in progress. If any loops are found, Scavenger breaks the loops and blinks the screen to show that a loop has been fixed. Afterwards, if changing the device is allowed, the Scavenger rebuilds the free list. This requires a write pass for all blocks on the free list and a write cursor is displayed. If any bad blocks were found, some more reads and writes are necessary to make the bad blocks into a well-formed chain. After this pass, Scavenger writes the new partition information block.

Next, the directory building pass is started. Scavenger asks whether it should rebuild the directory. Sometimes it recommends that you do this, otherwise there is no default. If you suspect there is a problem with the directories and if there are enough free blocks, answer "YES". The old directories will be deleted, if you so specify. Otherwise they are saved for later inspection. A "$" is appended to the end of their names and their file type is changed to ExDirFile (directories all have the type DirFile). New directories are then created whenever needed. This means that empty directories are not recreated. The old directories are just files that you can delete after the scavenge.

Note: This scheme makes it easy to recover from overwriting or deleting a directory since the directory reappears after a scavenge.

As described above, the Scavenger checks and allows fixing file lengths if desired. For each file, it checks the stored length with the actual number of blocks in the file. If they do not match, it allows you to specify a new stored length. This can be any value, but making it bigger than the number of blocks in the file is not recommended. The default for the stored length is the number of blocks in the file. The Scavenger does not check the lengths for directory files or files with their type field set to "SWAPFILE."

Each file has a table which points to each logical block of the file. This allows the file system to find a random logical block without searching down the chain from the file start. This table is called the "Random Index Table." Scavenger, as part of the directory building phase, can rebuild the random indices for all files. There is a separate question for this with a default answer of "NO". There is usually no reason to rebuild the indices unless Scavenger asks you to. Building the random indices for large files takes a long time.

If a bad file was created, Scavenger will ask for a name for the that file at the end of the directory building phase. If you allow Scavenger to enter and fix the indices for the bad file, you can then type and edit it as a normal file. In this way some useful information may be reclaimed.

5. MakeBoot.

The MakeBoot program creates new systems. An overview of its use appears in the manual "How To Make a New System". The "PERQ Introductory User Manual" describes the booting process. MakeBoot creates a boot file taking a stand alone run file (such as a system) and then associates this boot file with a letter. The lower case letters are assigned to the hard disk and the upper case letters are assigned to the floppy disk. The default that is used when no keys are held down is lower case "a". Boot letters can be freed of the associated boot by deleting the system and/or interpreter boot files.

Any program can be made to work "stand-alone" (so it can be booted) by initializing various modules as the System program does. It is generally not necessary or desirable to have programs other than the System be stand-alone.

The run file name given to MakeBoot determines on which device and partition the boot will be. MakeBoot takes the directory part of the file name and uses that to determine on which device and partition to put the boot. Therefore, to make a boot somewhere, first copy the run file to that partition. After specifying the run file, Makeboot asks for the configuration file. This file tells MakeBoot which System modules are swappable. The format of the file is:

```
Module-name  swappability
```

where swappability is "sw" for swappable, "um" for unmovable (stronger than unswappable), or "us" or blank for unswappable. The default is unswappable so only the modules in your system that you want to be swappable or unmovable need to be listed. The default system config file (named System.nn.Config) is:

```
*SAT* UM
*SIT* US
*Cursor* UM
*Screen* UM
*Font* US
*IO* UM
System SW
Stream SW
Writer SW
IOErrMessages SW
Loader SW
Reader SW
Perq_String SW
Screen SW
FileSystem SW
Code SW
GetTimeStamp SW
FileDefs SW
Memory SW
IO_Init SW
```

```
RunRead SW
FileDir SW
Scrounge SW
```

The system data segments that the hardware uses are required to be unmovable, the data used by software (*SIT* and *FONT*) are required to be unswappable, and everything else that is not used by the swapping system itself can be swappable.

The next question MakeBoot asks is whether to write the boot microcode onto the device. No matter how many boot letters are defined for a device, there is only one set of boot microcode so this only needs to be written when putting the first boot onto a device. The standard boot microcode files are "SysB" and "Vfy".

There are two files associated with each boot letter. The system boot file is Pascal and the interpreter boot file is microcode. The system boot file is created by MakeBoot by reading the supplied run file. The standard microcode is usually used with all boot files. It is used by MakeBoot to create the interpreter boot file if you so specify. If you have already created a boot for the current letter and you have not changed microcode, it is not necessary to make a new interpreter boot file, but it never hurts to do so. If you want to load the standard microcode and it is found by MakeBoot, type CR when it asks for an interpreter microcode file.

Included in the system code is the default character set font. MakeBoot looks for the default font (currently, "Fix13.kst") in all the search paths. If it is not found, you will have to supply a font file name.

Note: For the Editor and certain other programs to work, the default font must be fixed width and thirteen bits high and nine bits wide.

MakeBoot puts the output boot files wherever you specify but it is important that the interpreter and system boot files be in the same partition. The device and partition in which the boot file is created will be the default path after the boot. This means that there must be at least a "Login.nn.run" and a "Shell.nn.run" (where "nn" is the version number of the system run file), in the root directory of the partition. It doesn't matter if the boot files are in a subdirectory in the partition; the run files mentioned above must be in the Root directory.

The MakeBoot program will take a switch on the command line. If it is invoked with the /BUILD switch, all arguments are specified on the command line and the user is not asked any questions. The format for this switch is:

```
MakeBoot [<dir>]System.<nn>/Build <bootKey>
```

where <dir> is an optional directory, <nn> is the system version number and <bootKey> is the character to boot from. Makeboot then uses the default answers for all questions.

6. FixPart.

FixPart is an experimental program for fixing the Device and Partition information blocks. It is not recommended that customers try to use it without assistance. Unlike the other programs described above, FixPart is only partially automatic and can cause a lot of damage. Unfortunately, it is currently the only way to fix bad partition and device information blocks.

Note: It is very rare that the device and partition information blocks get broken, so Scavenger should always be run first to see if the problem is actually elsewhere.

FixPart first asks if you are sure you want to run the program. Next, it asks for the device type. It then goes through and checks each partition for consistency with the other partitions and with the device information block. If a name is dubious it asks if it is valid or not. After all partitions are checked, the program gives a summary of the errors. If none were found, then it exits; otherwise, you can specify new start and end addresses for the partition and fix the names.

If the device information block is not writeable, then you have to reformat the entire device and, unfortunately, lose all the data on it. If one of the partition information blocks is not writeable, then you may be able to save some information by using the partition program to join the partition with the bad information block to the partition before and then scavenging. If it is the first partition, however, your device will have to be reformatted.