The Helios Getting Started Tutorial

ACKNOWLEDGEMENTS
This manual was written by Bill Noble and Rachel Ganz of Vardas.

COPYRIGHT
This document is Copyright ©1990, Distributed Software Ltd. All rights reserved. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from Distributed Software Limited, The Maltings, Charlton Rd, Shepton Mallet, Somerset, BA4 5QE, UK.

TRADEMARKS
Helios is a trademark of Perihelion Software Ltd. DSL/// is a trademark of Distributed Software Ltd. UNIX is a trademark of AT and T.

EDITION
This edition first published October 1990.
Published by Distributed Software Ltd.

Printed in the UK.
Part number H09016
Edition 1

The Helios software is available from Distributed Software Limited.
The Maltings, Charlton Rd, Shepton Mallet, Somerset, BA4 5QE, UK, and from selected distributors worldwide.
INTRODUCTION

This tutorial is aimed at readers who are about to use Helios for the first time and have little or no experience of Unix-like systems. It provides a simple introduction to the most commonly used features of the Helios user interface.

The tutorial consists of a sequence of descriptions of commands with associated examples. The examples should be done in the order they are given. They are boxed to make them stand out from the descriptions. The bold text shows what you should type in and the lighter text shows what you should see on the screen. All actions in the tutorial are numbered consecutively.

⚠️ It is essential that you do all the examples in the correct order as they are interdependant.

Each time a new command is introduced its name appears in the margin together with a one line summary in the main column. This should help you to reference them quickly.
1.1 **TYPOGRAPHIC CONVENTIONS**

Throughout this tutorial the following typographic conventions have been used:

- *italic* Words appearing in *italic* mark a new term in the text. This is where the term will be defined.

- **screen text** Words appearing in the **screen text** font refer to command names, command lines, and any text which is displayed on a screen.

- **user text** Words appearing in the **user text** font refer to text that should be typed by the reader when doing one of the examples.

- **KEY** Words that appear within a **BOX** refer to keys on the keyboard.

- **X|Y** Means key **X** should be pressed and then key **Y**.

- **X+Y** Means key **X** should be pressed at the same time as key **Y**.

---

The Helios Getting Started Tutorial

This tutorial assumes that you have correctly installed your copy of Helios.

Some of the practicals assume you are running Helios with a PC hosted system.
This chapter provides a brief introduction to the most commonly used features of the Helios user interface.

### 2.1 STARTING HELIOS

Before trying the examples in this tutorial you must start Helios.

1. Make the `\helios` directory the current directory. E.g.,

   ```
   C: cd \helios RETURN
   C:
   ```

2. Start Helios by executing the `server` program:

   ```
   C: server RETURN
   ```

   The `server` program loads Helios onto the transputer board.

You should now have a running Helios system (if not, you should check carefully that you have installed Helios correctly).
The information displayed on your screen will look something like this:

```
console
Helios Operating System
Version xx
(C) Copyright 1987-90, Perihelion Software Ltd.
```

**login:**

Helios will be waiting for you to *login*. Logging in is the process by which you tell Helios who you are. This is described in more detail in the next chapter.

The default Helios system knows about a user called *guest*. You should use this name when logging in for the tutorial.

⚠️ If Helios does not ask you to login it means your system has been set up differently. In this case you should consult the person responsible for installing the Helios system before continuing with this tutorial.

3. Login to Helios as ‘guest’:

```
login: guest [RETURN]
```

```
Welcome to the Helios Operating System
```

If Helios does not recognise the login name you give it will first ask for a password and then display the message “Login incorrect”. You will then have the opportunity to login again.

The ‘%’ character displayed on the line after the welcome message is the Helios *command-line prompt*. This tells you that Helios is ready to receive a command. When a command has been executed Helios responds with another command-line prompt.
2.2 **TYPING, EDITING AND EXECUTING COMMANDS**

All commands issued to Helios at the command-line prompt, are *interpreted* (i.e., understood and acted upon) by a part of the Helios operating system called the *shell*. The shell is a program which interfaces between you and the operating system. The Helios shell has been designed to look as similar to the standard Unix shell (csh) as possible.

⚠️ **Users who are familiar with the Unix shell should have no difficulty in using Helios. However you should note that some Unix shell commands are not available under Helios, and that some commands with the same name may not behave in exactly the same way.**

Commands are given to the shell by typing the command name, together with any parameters, at the command-line prompt (‘%’). When you press the [RETURN] key the shell program interprets the command and causes the appropriate program to run.

You can interrupt or cancel a command that has just been issued by pressing [CTRL]+[C] (i.e., the control key and the ‘C’ key pressed together).

---

The simplest commands consist of a single lower-case word. For example:

```
To display today’s date:

4 % date [RETURN]
Date : Tue Sep  4 10:00:22 1990
%
```

If you make a mistake when typing a command name you will get an error message of the form:

Command not found

For example:

```
Entering an incorrect command:

5 % fate [RETURN]
fate: Command not found.
%```
EDITING THE COMMAND LINE

If you notice a mistake before pressing the RETURN key you can correct it by deleting back to the point where the error was made and retyping the rest of the command. The labeling of the delete key will vary between makes of keyboard. The delete key is usually one of:

[BACKSPACE], DELETE, DEL, RUBOUT, ←

Alternatively you can use the control sequence [CTRL]+H to delete the previous character on the line.

Many Helios commands take parameters which affect what the commands do. The format of a command with parameters is:

command-name option(s) filename(s)

This means:

a command name followed by none or more options, followed by none or more file names. These terms are described below.

command-name

All Helios commands should be entered in lower case. For example, 'Date' is not the same as 'date'. If you type 'Date' Helios will respond with the message "Command not found".

option(s)

The options affect the way the command works. Options are normally prefixed by the minus character ‘-’. Option names are mostly single letter and are case sensitive. This means that option ‘-t’ is different from option ‘-T’. If a command has more than one option the options can either be typed separately, each preceded by the ‘-’ character, or combined with just one minus sign preceding the lot. For example,

-t -s or -ts

filename(s)

Many commands take information from a file and manipulate it in some way before producing some output. The filename parameter(s) specify which file(s) are to be processed.

You must put a space between the command name and the options, and between the options and the file names.
LISTING THE FILES IN A DIRECTORY

An example of a command with parameters is the listing command, ‘ls’, which lists the contents of a directory.

The `ls` command without parameters produces a simple listing of the files within the current directory:

```
6 % ls
   cshrc
   examples/
   login
```

This generates an alphabetical listing of the contents of the guest directory.

The `ls` command with the `-l` (long) parameter produces a detailed listing of the files in the directory:

```
7 % ls -l
   f rwe---da 0 128 Fri Aug 31 12:53:20 1990 cshrc
   d rwvxyzda 0 0 Mon Sep 3 09:17:38 1990 example/
   f rwe---da 0 1 Thu Aug 9 10:04:12 1990 login
```

The ‘-l’ option tells `ls` to provide a line of information on each file. Amongst the information listed is whether something is a file or a directory, the size of the file and the time and date the file was last altered. The first column of the listing indicates the type of file. ‘d’ is a directory file and ‘f’ an ordinary file such as a data file or executable program. Following the type of file is the access mode which specifies who is allowed to read (r), write (w) or execute (e) the file.
The `ls` command with a filename as a parameter produces a listing for that file:

```
To display detailed information on the file 'cshrc':
8 % ls -l cshrc [RETURN]
  rwe---da 0 128 Fri Aug 31 12:53:20 1990 cshrc
%
```

In this example the listing is limited to the file 'cshrc'.

### ONLINE HELP INFORMATION

There will be times when you cannot remember what parameters a particular command takes. When this happens you can use the Helios on-line help facility which provides detailed information on every Helios command.

### The Helios Getting Started Tutorial

Here is an example of the `help` command:

```
To display help information on the `printenv` command:
9 % help printenv [RETURN]
  printenv: Displays environment variables

  Format: printenv

  Description: The printenv command is used to display the names and values of all environment variables that are currently set.

  See also: `setenv`, `unsetenv`

Quit  ?Help  Go back
Q%
```

To exit from the help program you must hit the `Q` key.
The help facility is much more powerful than this example might imply. You can find out more about help simply by typing:

```
help
```

**COMMAND-LINE HISTORY – REISSUING OLD COMMANDS**

Helios keeps a record of the commands that are issued at the command-line prompt. You can browse through this history using the cursor up and down keys (↑, ↓).

Pressing the ↑ key causes the previously executed command line to be displayed at the command-line prompt. You can then re-issue this command by pressing the RETURN key. Each press of the ↑ key displays an earlier command line which can be re-issued, or not, as you wish. Helios has a limit to the number of command lines it can remember (this is set at 20 in the default system and can be modified by editing the cshrc file (see later)).

The ↓ key scans backwards through the command-line history whilst the ↑ key scans forwards.

You can re-issue the previously issued command by typing ↓↑RETURN.

---

**The Helios Getting Started Tutorial**

To re-display and re-issue the previous command line:

```
10 % ↑
  % help printenv RETURN
  <More text>
  %
```

After pressing the ↑ key the cursor is positioned at the end of the line to allow you to re-issue the command.

**Examining the command-line history**

Play with the ↑ and ↓ keys to browse backwards and forwards through the command-line history.

**DISPLAYING A LIST OF RECENTLY EXECUTED COMMANDS**

Helios keeps a record of previously executed commands. In the default system the last twenty commands that you typed are recorded. You can display this list using the history command, which takes two parameters:

```
history [-r] [<n>]
```
Where:

[-r] is an option which causes the list to be displayed in reverse order.

[<n>] is an optional number which specifies that only the last <n> command lines should be displayed.

The shell variable ‘history’ specifies the maximum number of command lines displayed by the history command. Shell variables are described in the next chapter.

To display a list of previously issued command lines:

```
12 % history [RETURN]
  1 date
  2 fdate
  3 ls
  4 ls -1
  5 ls -1 cshrc
  6 help printenv
  7 help printenv
  8 history
%
```

The command lines displayed by history are numbered. You can use these numbers to selectively re-issue the command lines of your choice. The command,

```
! <n>
```

will re-issue the command line number <n>

To re-issue the ‘date’ command (which was the first command issued in this tutorial):

```
13 % !1 [RETURN]
date
date : Tue Sep 4 10:02:12 1990
%
```
2.3 FILES AND THE FILING SYSTEM

Helios files are kept in places called directories. Directories are hierarchical storage areas that resemble upside-down trees in structure.

```
    root
     ↓
    trunk
      ↓
   branch        another branch
       ↓   ↓
      twig1 twig3
       ↓   ↓
      twig2
```

Each directory has a name, in the same way as files have names. A directory is a file containing information about the whereabouts of the files in that directory.

Every file is located in a unique directory. Directories can contain sub-directories. You can think of a directory as being a folder which can contain files and other folders.

In the diagram above ‘trunk’ is a directory containing the directories ‘branch’ and ‘another branch’.

A user of Helios is always considered to be in a directory known as the current directory. Which directory is the current directory will depend on what your default login directory is and whether or not you have changed directory since starting Helios (for information on login directories see the next chapter).

Helios uses the current directory as the default source and destination for files used by the various Helios commands. This means that Helios will always look for the files you are using in the current directory, unless you specify otherwise (see below).
When you first ran the Helios system at the start of this tutorial the current directory was guest. Let's have another look at this directory:

To display a list of the files in the current directory (guest):

```
% ls
```

```
cshrc  examples/  login
```

The guest directory contains a sub-directory called examples. The trailing ‘/’ character indicates that this is a directory name.

To display the contents of the sub-directory examples:

```
% ls examples
```

```
convol/  factor/  hello/  lb/
pi/  tut/
```

Note that the parameter to the listing command `ls` was 'examples' and not 'examples/'. When referring to a directory you can omit the trailing ‘/’ character.

The Helios Getting Started Tutorial

The following illustration shows a simplified view of the directory structure for the standard Helios installation:

```
/ (the root directory)
  
  00 10
  
  helios
  
  bin  etc  users  lib
  
  guest  root  shutdown
  
  examples
```

At the very top of the directory structure is the root directory called ‘/’. The root directory contains two sub-directories called 00 and 10. The IO directory contains a sub-directory called helios and so on.

By the end of this tutorial you will have created your own directory area which will be the default current directory when you next run Helios. This is called the login directory and is normally unique for different users.
Directories are identified using path names. A path name specifies the unique path through the directory structure to the directory you want. Path names consist of a sequence of directory names separated by the ‘/’ character.

There are three ways of giving the path name for a directory:

1. **Relative to the root directory.** This is known as the *absolute path name* because it will always identify the correct directory no matter what the current directory is (hence the term ‘absolute’).

2. **Relative to the current directory.** This is known as the *relative path name*. The relative path name will only work if it is correctly specified relative to the current directory. Relative path names are often more convenient because fewer characters have to be typed in for the path name. If the current directory is changed the relative path name for a particular directory should also change.

3. **Relative to an alias server.** This is described in the next chapter.

The absolute path name for the guest directory is '/IO/helios/users/guest'. Similarly, the absolute path name for the examples directory is '/IO/helios/users/guest/examples'.

If the current directory is guest the relative path name for the examples directory is 'examples'. If the current directory is /O the relative path name for examples would be 'helios/users/guest/examples'.

The first character of the path name tells Helios what sort of path name is being used:

- **<name>** A path name starting with a <name> is relative to the current directory.
- **../** A path name starting with ‘../’ is relative to the directory containing the current directory. For example, if the current directory is /IO/helios/users/guest/examples then ../cshrc refers to the cshrc file in the guest directory.
Using Helios

`~` A path name starting with `~/` is relative to the login directory (see the next chapter for information about login directories).

`/` A path name starting with `/` is an absolute path name.

**pwd**

**DISPLAYING THE ABSOLUTE PATH NAME FOR THE CURRENT DIRECTORY**

Helios provides a command called `pwd` which displays the absolute path name of the current directory. This can be useful for checking if a relative path name is correct relative to the current directory.

To find out the absolute path name of the current directory:

```
16 % pwd RETURN
/IO/helios/users/guest
```

**cd**

**CHANGING DIRECTORY**

All the examples so far have been relative to the default login directory `guest`. To move to another directory (i.e., to change the current directory) use the `cd` command.

To move to the root directory:

```
17 % cd / RETURN
```

To check that the current directory is the root directory:

```
18 % pwd RETURN
/
```

To move to the IO sub-directory using a relative path name:

```
19 % cd IO RETURN
```

Don’t forget to use capitals for ‘IO’. Directory names, like file names and command names, are case sensitive.
To move to the `examples` sub-directory using an absolute path name:

```bash
20 % cd /IO/helios/users/guest/examples
```

If you use `cd` on its own (i.e., without giving it a directory name) the current directory will become the login directory (or whatever the shell variable `HOME` is set to – see next chapter).

---

**CREATING NEW DIRECTORIES**

You can create new directories using the `mkdir` command. Let's create a directory called `red`:

```bash
21 % cd ..
```

Create a new directory called `red`:

```bash
22 % mkdir red
```

This creates a new sub-directory within the `guest` directory called `red`.
Display a list of files to check that the new directory has been created:

```
23 % ls
    cshrc examples/
    %
```

FILE NAMES
There are several ways you can give a file name parameter to an application.

- The file name alone
- The complete path name of the file
- A file name using wildcards
- File name completion

1 The file name alone

If you just give the name of the file, Helios assumes that the file is in the current directory. For example ‘cshrc’ will correctly identify the cshrc file if the current directory is /IO/helios/users/guest. Many file names contain two parts separated by the character ‘.’.

2 The complete path name of a file

A more precise way of identifying a file is to give the complete path name of the file. The path name can be either relative to the current directory or relative to the root directory. For example (assuming that the current directory is /IO/helios/users):

- guest/cshrc
  Uses a path name relative to the users directory to identify the file cshrc within the guest sub-directory.

- /IO/helios/users/guest/cshrc
  Uses an absolute path name to identify the file called cshrc.
A file name using wildcards

Helios provides a shorthand method called wildcards for identifying files or groups of files. There are three types of wildcards which, if used within a file name, cause Helios to fill in the file name for you.

The examples below assume that the current directory is

/IO/helios/users/guest/examples/tut

which contains the files:

testa1
testa2
testb1
testb2
testc1
testc2

The three types of wildcard are:

* If you put a '*' in a file name, Helios will attempt to replace the '*' with any sequence of characters which results in a valid file name. A valid file name is the name of an existing file in the current directory (or the directory given by the path name). Helios will generate a filled-in file name for every valid solution. For example (assuming the current directory is tut):

test* will expand to testa1
and testa2
and testb1
and testb2
and testc1
and testc2

xyz* will generate the error message "No match"

? If you put a '?' character in a file name, Helios will attempt to replace the '?' with any single character which results in a valid file name. For example (assuming the current directory is tut):

test?1 will expand to testa1
and testb1
and testc1

? will generate the error message "No match"
You can restrict which characters can be used in a substitution by putting the allowed characters within square braces. Helios will try to make a valid file name which includes one of the specified characters at that point in the file name. For example (assuming the current directory is \texttt{tut}):

\begin{verbatim}
\texttt{test[ac]l} \quad \text{will expand to} \quad \texttt{testal}
\hspace{1em} \text{and} \quad \texttt{testcl}

test[xyz]l \quad \text{will generate the error message "No match"}
\end{verbatim}

4 \quad \textbf{File name completion}

At any time, whilst typing in a file name at a command line, you can ask Helios to have a go at filling in the rest of the name for you. If the characters you have typed so far are enough to uniquely identify a file name you can press the \texttt{[ESC]} key and Helios will fill in the rest of the name. If what you have typed so far does not uniquely identify a file name, Helios 'beeps' complainingly and puts in as much of the file name as it can (i.e., any characters which are common to all the files which start with the string of characters you typed). If you type \texttt{[CTRL]+D} Helios will list all the names that match.

You should take care not to type \texttt{[CTRL]+D} on an empty command line as this may result in the current shell being terminated.

For example, assuming the current directory is \texttt{/helios}:

\begin{verbatim}
1s h[ESC] \quad \text{will expand to} \quad 1s host.con
\hspace{1em} \text{because there is only one file beginning with 'h'}.

1s l[ESC] \quad \text{will expand to} \quad 1s l[\text{BEEP}!]
\hspace{1em} \text{because there is more than one sub-directory beginning with 'l'}.

1s li[ESC] \quad \text{will expand to} \quad 1s lib[\text{BEEP}!]
\hspace{1em} \text{because there is only one file or sub-directory name beginning with 'li'}
\end{verbatim}

\texttt{[ESC]} can also be used to fill-in path names and command names.
Now try some of the different ways of identifying files for yourself:

Move to the `tut` directory:

```bash
$ cd examples/tut
```

To list some file names using the `*` wildcard:

```bash
$ ls *a
```

```
testa1   testa2
```

To list all the file names using `*`:

```bash
$ ls *
```

```
testa1   testa2   testb1   testb2
testc1   testc2
```

To list a filename using the `?` wildcard:

```bash
$ ls test?2
```

```
testa2   testb2   testc2
```

To list a filename using a restricted wildcard `[@]`:

```bash
$ ls test[ac]1
```

```
testa1   testc1
```
DISPLAYING THE CONTENTS OF A FILE

The simplest way of displaying the contents of a text file on your screen is to use the `more` command. For example:

<table>
<thead>
<tr>
<th>To display the contents of the cshrc file:</th>
</tr>
</thead>
<tbody>
<tr>
<td>29 <code>more /IO/helios/users/guest/cshrc</code></td>
</tr>
<tr>
<td>set history=20</td>
</tr>
<tr>
<td>set savehist=20</td>
</tr>
<tr>
<td>alias h history</td>
</tr>
<tr>
<td>alias ls ls -F</td>
</tr>
<tr>
<td>alias me emacs</td>
</tr>
<tr>
<td>set path=( ./helios/local/bin /helios/bin )</td>
</tr>
</tbody>
</table>

The `cshrc` file contains a shell script which is executed every time you run a shell. More information on this is provided in the next chapter.

The contents of a file are often too long to be shown on the screen at the same time. `more` will automatically split the displaying of long files into chunks the size of the display screen. After each page is displayed the message "—More—" appears at the bottom of the screen. `more` will then pause until you press one of the following keys:

- **SPACE** Displays the next page-sized chunk of the file.
- **RETURN** Displays the next line of the file.
- **Q** Quits displaying the file.

An alternative way of controlling the way a file is listed on the screen is by pressing the **CTRL+S** (stop) and **CTRL+Q** (quit stop) keys. Pressing **CTRL+S** causes Helios to stop displaying any further information on the screen until you press **CTRL+Q** (quit stop). This means you can use **CTRL+S** and **CTRL+Q** to hold the display while you read it.
Try using `more` to display the file `emacs.hlp` in the `/IO/helios/etc` directory:

```
To page through the `emacs.hlp` file using `more`:

% more /IO/helios/etc/emacs.hlp
=>
   MicroEMACS 3.8 Help screens
   (01/18/87)
   M- means to use the <ESC> key prior to using another key
   ^A means to use the control key at the same time as the A key
   ^V or [Pg Dn]  Scroll down
   ^Z or [Pg Up]  Scroll up
   ^C or <INSERT> Insert a space

The cursor is left at the end of the line "--More--". Nothing will happen until you press an appropriate key. Experiment with the [RETURN] and [SPACE] keys to page through to the end of the `emacs.hlp` file. If you get bored you can quit by pressing [CTRL]+[C].
```

---

**CREATING AND EDITING TEXT FILES**

Text files can be created and edited using the `emacs` text editor supplied with Helios. In this tutorial we will only show the simplest of `emacs` operations. For fuller details you should consult "The MicroEmacs Editor: A Guide for Beginners".

`emacs` takes the name of the file you want to edit as a parameter. If the file already exists you will be able to edit it. If the file doesn't already exist it will be created.

When `emacs` runs, it displays a banner at the bottom of the screen showing the name of the file which is being edited. For a newly created file the rest of the screen will be blank and the cursor will be positioned at the top left-hand corner of the screen. You can then type the text of the file. If you make any mistakes you can use the cursor control keys (up, down, left and right) and the delete key to correct the mistake.

When you have finished entering the file you can save it and exit from `emacs` by pressing [ESC]+[Z].

If you want to quit from `emacs` without saving the file you should press [CTRL]+[X] followed by [CTRL]+[C]. `emacs` will remind you that you will lose the edits you have made by displaying the message:

"Modified buffers exist. Leave anyway [y/n]?"

If you really are happy about losing the edits then you should press the [Y] key.
Here is how to create a simple two line text file:

First move to the red directory that you created earlier:

```
31 % cd /IO/helios/users/guest/red  RETURN
```

To create a two line file called rose:

```
32 % emacs rose  RETURN
```

What's in a name? that which we call a rose  RETURN
By any other name would smell as sweet  RETURN
ESC Z

[Saving rose]
```

DUPLICATING FILES

You can make a duplicate copy of a file using the copy command, cp, which can take two parameters:

cp <source-file> <duplicate-file>

To make a duplicate copy of the file rose called juliet:

```
33 % cp rose juliet  RETURN
```

To confirm that juliet is a copy of rose:

```
34 % more juliet  RETURN
```

What's in a name? that which we call a rose
By any other name would smell as sweet
```
MOVING AND RENAMING FILES

If you want to move a file from one directory to another you should use the move command, `mv`, which takes two parameters:

`mv <source-file> <destination-file>`

By choosing a different name for the `<destination-file>` the `mv` command can be used to rename files.

To move the file `juliet` to the guest directory:

```
% mv juliet ./juliet
% ls ..
```

```
cshrc examples/ juliet login
red/
```

To rename the file `rose`:

```
% mv rose quote
% ls
```

```
quote
```

The `ls` command confirms that the current directory now contains the single file `quote`.
**Removing/Deleting Files and Directories**

Directories and files can be removed (deleted) using the `rm` command, which takes as its parameter the names of the files you want to remove:

```
rm <file-names>
```

First let's delete the file `juliet` in the `guest` directory:

```
Move to the guest directory:
37 % cd ..  RETURN
%
Delete the file juliet:
38 % rm juliet  RETURN
%
```

When deleting a directory with `rm` you must take account of whether or not the directory has any files in it. If you want to delete a directory that contains files or subdirectories you must either delete all the files and subdirectories first, or use the `rm '-r'` option, which tells `rm` to delete all subfiles and directories.

⚠️ You should be very careful about using the `'-r'` option as it is possible to do a great deal of accidental damage this way (for example, the command `rm -r /helios` would delete the entire Helios installation).

To delete the directory `red` that you created earlier:

```
Check you are in the correct directory:
39 % ls  RETURN
cshrc examples/ login red/
%
Remove the directory red and everything it contains:
40 % rm -r red  RETURN
%
```
Check that the red directory has been deleted:

```
41 % ls
  cshrc     examples/
  login
```

**logout**

QUITTING HELIOS
To quit from Helios simply type ‘logout’ at the command-line prompt:

```
To quit from Helios:
42 % logout
C:
```

If you ran Helios without logging in you can quit using the `exit` command.

ADVANCED USE OF HELIOS

This chapter introduces some of the more advanced features of Helios. To keep this tutorial short the descriptions below have been kept as brief as possible. The aim is to give you a feel for what can be done rather than explain everything you need to know. For more information you should consult “The Helios Operating System” manual.

If you have been following all the examples in this tutorial you will now need to restart Helios.
To restart Helios and login as guest:

```
C: cd \helios
C: server
```

```
login: guest
Welcome to the Helios Operating System
```

In this chapter, amongst other things, you will be shown how to create your own login name and login directory.

To create and move to your own working directory:

```
% cd /O/helios/users
% mkdir myname
% cd myname
```

This creates a new directory called myname and makes it the current directory.

3.1 **MORE ABOUT FILES**

**STANDARD INPUT AND OUTPUT**

Unless you specify otherwise, Helios commands expect their input to come from the keyboard and their output to go to the screen. This is because, by default, the programs read from the *standard input device* (called stdin) and write to the *standard output device* (called stdout). stdin is normally connected to the keyboard and stdout to the screen.
CONCATENATING FILES

cat copies data from the specified files to the standard output. If you omit to give cat a file name it will try to read from the standard input device (normally the keyboard). If you give cat more than one file name it will copy each of the files to the output (in effect appending one file onto the end of the other, cat is short for concatenate). The format of a cat command is:

cat [-nsvbt] [<file> ...]

The ‘-nsvbt’ bit is a list of the options (for example ‘-n’ means: number each line of output).

REDIRECTING INPUT AND OUTPUT

You will often want a program to take its input from a file (not the keyboard) and write its output to another file (not the screen). You can redirect the standard input and standard output channels by using redirection characters. The redirection characters are ‘<’, ‘>’, and ‘>>’. Here’s how they work:

< is used to redirect stdin. The sequence

< filename

means: redirect the standard input so that it comes from the file called filename. For example:

cat < quote

causes cat to take its input from the file quote instead of from the keyboard.

>> also redirects stdout. It is similar to ‘>’ except that the sequence

>> filename

means: redirect the output so that it is appended to the file called filename. For example:

cat >> quote

causes the output of cat to be appended onto the end of the file called quote. As before, the input for cat will come from the keyboard.
The following examples show how redirection can be used to alter how cat works:

To use cat to create a file:

```
45 % cat > quote  RETURN
   He who can does,  RETURN
   He who cannot teaches.  RETURN
   CTRL+D
```

In this example cat takes its input from the keyboard (because no files were specified in the command line) and writes its output to the file 'quote'. The CTRL+D at the end of the input tells Helios that this is the end of the input text.

To check that the file has been created correctly:

```
46 % cat quote  RETURN
   He who can does,  RETURN
   He who cannot teaches.  RETURN

This outputs the named file to stdout (the screen).
```

To append some text onto the end of a file:

```
47 % cat >> quote  RETURN
   [G. B. Shaw]  RETURN
   CTRL+D
```

To display the new file:

```
48 % cat < quote  RETURN
   He who can does,  RETURN
   He who cannot teaches.  RETURN
   [G. B. Shaw]
```

In this example the '<' redirection character was used to connect the standard input to the file 'quote'. This is not a very good example because it would be better to call cat with quote as a file name parameter.
Advanced Use of Helios

PRINTING FILES

Redirection can also be used to print files. Helios assumes that the printer is connected to a default device called /printers/default. To print a file you simply copy it to /printers/default. You can do this with the cat command or the cp command. It is better however to use the pr command which is provided specifically for this purpose. pr formats the file as it sends it to the printer. This means it splits the file into pages of output and puts the date, time and file name at the top of the page. The output of pr is sent by default to stdout, so if you want it to go to the printer you must specifically redirect it.

⚠️ The following two examples will only work if you have a printer connected to /printers/default. (Make sure it is turned on!)

⚠️ If you are not using a PC compatible host computer this method of printing may not work. For more information consult the release notes for your machine.

To print a file:

```bash
% pr quote > /printers/default RETURN
```

---

3.2 PIPES, BACKGROUND JOBS AND REMOTE TASKS

PIPES (‘|’) – CONNECTING THE OUTPUT OF ONE PROGRAM TO THE INPUT OF ANOTHER

Sometimes you will want the output from a program to be used as the input of another program. For example, the output of the C compiler is normally used as the input of the assembler. Helios provides a mechanism called piping which automatically connects the stdout channel of one program to the stdin channel of another. This removes the need for temporary intermediary files.

The following example shows piping the output of ls to the input of more (to enable paging of very long directory listings).

⚠️ On some keyboards the pipe symbol ‘|’ appears as a broken vertical bar.
Advanced Use of Helios

To pipe the output of ls to the input of more:

```
% ls -l /IO/helios/bin | more RETURN
```

61 entries
f rwe---da 0 2516 Fri Apr 7 15:39:22 1989 ascii
f rwe---da 0 55820 Mon Jan 15 12:53:12 1990 asm

(more text)

f rwe---da 0 1376 Wed Aug 8 15:44:54 1990 dump

--More--

more will wait for you to press a key (either SPACE to display the next page, or Q to quit).

To display the next page of the listing:

```
--More-- SPACE
```

(more text)

f rwe---da 0 1376 Wed Aug 8 15:44:54 1990 dump

(more text)

f rwe---da 0 1216 Wed Aug 8 15:44:54 1990 pr

--More--

The Helios Getting Started Tutorial

To quit from more:

```
(more text)
```

f rwe---da 0 1216 Wed Aug 8 15:44:54 1990 pr

--More--

BACKGROUND JOBS

You don’t always have to wait for a command to finish before continuing with the next one. Helios allows you to execute jobs in the background. This means that the Helios shell will continue to respond to your commands whilst Helios handles all the jobs which are being run in the background. A command becomes a background job when you put an ampersand (‘&’) on the end of the command line.

After launching a background job, Helios displays a job number together with a process identification number (see “The Helios Operating System” manual for more information on process identification numbers). The job number is unique for every job in the system and allows you to identify the job to commands such as kill (described below).

When a background job finishes, Helios displays a message of the form:

```
[<job-number>] Done <command-line>
```

60
Advanced Use of Helios

<job-number> is the number of the job that has just finished and <command-line> shows the command which was executed.

Background jobs are useful if you have a task which is likely to take a long time and requires no user interaction. For example, if you have a very long file that you want to print you could set it printing in the background and carry on with the next task (such as editing another file).

⚠️ Some background jobs, such as the print example below, can take up a lot of processor time. If you have a single processor system you may find some background jobs slow the system down so much that there is no significant advantage in executing them in background mode.

To print the emacs.hlp file in background mode:

```
54 $ pr /IO/helios/etc/emacs.hlp > /printers/default & RETURN
[1] 18
%
```

The first number (in square brackets) is the job number. The other is a process identification number. You get the command-line prompt back straight away, even though the printing has not finished.

Listing the active background jobs

You can display a list of the currently active background jobs using the jobs command. This displays the job number, status (i.e., whether or not the job is running), and original command line for each active background job.

```
To display a list of currently active background jobs:

55 $ jobs RETURN
[1] + Running pr /IO/helios/etc/emacs.hlp
%
```

The jobs command displays the job number, status, and original command line for each active background job. In this example there is only one job running.

Killing a background job

You can kill (abort) a background job using the kill command together with the job number of the job you want to abort. This number was displayed when you launched the job (and can also be seen if you issue a jobs command).
To kill the background job with job number 1:

```bash
% kill %1
```

Terminate signal

The '%' before the '1' tells kill that the number is a job number (no '%' means it is a process identification number).

---

**RUNNING A PROGRAM ON ANOTHER PROCESSOR**

If you have a multi-processor system, Helios will automatically allocate programs to the various processors (see "The Helios Parallel Programming Tutorial"). If you want to control the allocation of programs to processors yourself you can use the `remote` command to explicitly request that a particular program is run on a particular processor. `remote` takes two parameters:

```
remote <processor-number> <command-line>
```

The command,

```
remote 02 ls /
```

will run the program to list the contents of the root directory (`ls /`) on processor number 02.

---

**DISPLAYING A LIST OF PROCESSORS AND TASKS**

The `ps` command displays a list of all the processors in your system together with a list of the tasks running on each processor.

```
% ps
```

To display a list of processors and tasks:

```
% ps
```

This example shows a typical listing for a single processor system. The processor is called ‘00’ and has 5 tasks running on it. Amongst these are the `shell` program and the `ps` program that generated this listing. The numbers on the ends of the names are task identification numbers.

For further information on using systems with more than one processor consult "The Helios Parallel Programming Tutorial".
3.3 **THE HELIOS SHELL**

The Helios shell is a program which deals with the commands typed at the command-line prompt '>'. It interprets the commands and causes the appropriate program(s) to run. It is the interface between you and the Helios operating system. You can run Helios with more than one shell if you want.

**shell**

**CREATING ANOTHER SHELL**

The Helios shell command creates a new copy of the shell interpreter. The old shell still exists but you won't be able to access it again until you exit from the new shell. The shell command is normally executed automatically when you log on.

**wsh**

**CREATING ANOTHER WINDOWED SHELL**

The wsh command can be used instead of the shell command. wsh creates a window for the new shell so that all the shells can be used. To switch between shells you must switch between windows. A window consists of an area on the screen which is dedicated to the shell. It displays the command-line prompt and everything that is sent to stdout (the screen) by programs running within that shell. How the windows are displayed will vary according to what system you are running.

If you are using Helios with a windowing package such as X windows, each wsh command will create a new window on the screen (which you can move, resize and overlap with other windows). You can switch between windows by moving the cursor (with a mouse) to the window of the shell you want to use.

**LOGGING IN AND LOGIN DIRECTORIES**

Many Helios systems will have more than one user. Each user will probably have their own directory space. Helios provides a mechanism for users to identify themselves called logging in. When a user logs in, Helios moves to a user specific login directory and runs a user specific program (which is normally the shell command to start the shell interpreter).
When you start Helios by executing the server program on the host computer, the last thing the server does is execute the login command. This command looks for a file called passwd in the /IO/helios/etc directory. This is where Helios keeps a record of user names, login directories (the directory that will be the current directory when you log in), and start programs (the program which is automatically run when you log in – normally the shell program). Helios will ask you for a name and, if required, a password. If those match those in the passwd file your login directory will become the current working directory.

The login command can be issued at the Helios command line just like any other command. This effectively causes the current user to be logged off and replaced with a new user.

The password part of the login procedure is optional. If your entry in the passwd file does not contain a password the login command will not ask you for one.

Let’s have a closer look at the information held in the passwd file. Each user entry is one line long and has the following format:

```plaintext
<login name> : <password> : <user #> : <group #> : <user name> : <login directory> : <start program>
```

Where:

- `<login name>` is the name you give when logging in. Typically this is your first name but could be any other name you like.

- `<password>` is an optional password.

- `<user #>` is a user number.

- `<group #>` is a group number.

- `<user name>` is the user’s name (a fuller version of the login name).

- `<login directory>` is the directory which will become the current directory when the user logs in.

- `<start program>` is the program which will be executed when the user logs in. This will normally be the shell command as most users will require a shell. It is possible to restrict certain users to specific activities by putting another program name here (for example /helios/bin/emacs would restrict the user to using emacs).

The default Helios passwd file looks like this:

```plaintext
root::1:0::/helios/users/root:/helios/bin/shell
shutdown::2:0:shutdown:/helios/users/shutdown:/helios/bin/shell
guest::100:0:guest:/helios/users/guest:/helios/bin/shell
```
In this file there are three users called root, shutdown and guest. The guest entry in the passwd file defines a user as follows:

- **Login name**: guest
- **Password**: <none>
- **User number**: 100
- **Group number**: 0
- **User name**: guest
- **Login directory**: /helios/users/guest
- **Start program**: /helios/bin/shell

You are now ready to reconfigure the Helios system to recognise you as a new user. In the examples below you should replace the words in italic with your first name and full name as appropriate.

To move to the /IO/helios/etc directory:

```bash
58 % cd /IO/helios/etc RETURN
```

To edit the passwd file using emacs:

```bash
59 % emacs passwd RETURN
```

```
root::1:0::/helios/users/root:/helios/bin/shell
shutdown::2:0:shutdown:/helios/users/shutdown:/helios/bin/shell
guest::100:0:guest:/helios/users/guest:/helios/bin/shell
name::99:0:full-name:/helios/users/mynname:/helios/bin/shell RETURN
ESC Z
```

Your login directory will now be /helios/users/mynname. When you login the first command to be executed will be shell.
To login as yourself:

```bash
% login
login: first-name
```
Welcome to the Helios Operating System

To check that your login directory has been set properly:

```bash
% pwd
/IO/helios/users/myname
```

### 3.5 CUSTOMISING YOUR SHELL

#### alias

**GIVING A COMMAND ANOTHER NAME**

If you don’t like the names that have been given to some of the Helios commands you can effectively rename them using the `alias` command. `alias` takes two (optional) parameters:

```
alias <new-name> <list-of-words>
```

Where:

- `<new-name>` is the new name you want the command to be known by.
- `<list-of-words>` is the command line you want to rename. This can be just the command name, or the command name together with some of its options and/or parameters.

⚠️ You should take care when choosing an alias that the name does not clash with an existing command name.
alias can be used to create shorthands for long commands that you use often. For example:

To create an alias called dir for the ls -l command:

```
% alias dir ls -l
```

To use the newly created alias:

```
% dir
3 Entries
  f rewe---da  0 56 Fri Jun 8 11:17:36 1990 quote
```

If you issue the alias command without any parameters it displays a list of all the currently active aliases.

```
% alias
```

The format of the command to alias a directory is:

```
/helios/lib/alias <name> <directory>
```

You can remove an alias using the unalias command.

```
% unalias dir
```

To check that the alias has been removed:

```
% alias
```

ALIASING OF DIRECTORY NAMES

In addition to creating aliases for commands you can also create aliases for directory names. This is particularly useful when you need to refer frequently to a long directory name. Directory aliases are created with a program called the alias server which must be run in background mode.

```
/helios/lib/alias <name> <directory>
```
Where:

/helios/lib/alias is the name of the alias server program.

<name> is the new name for the directory.

<directory> is the full path name for the directory to be aliased.

& causes the command to be executed in background mode.

When you use a directory alias you must prefix the directory name with a '/ '(e.g., /helios and not helios).

⚠️ The alias server is not the same as the alias command. The alias server is used for directories and the alias command is used for commands (which can only be accessed from the shell).

---

**The Helios Getting Started Tutorial**

To create an alias for the /IO/helios/users/guest/examples directory:

```
67 % /helios/lib/alias examples /IO/helios/users/guest/examples &
```

To list the examples directory using the /examples alias:

```
68 % ls /examples
convol/  factor/  hello/  lb/
pi/      tut/
%
```

**SETTING THE SHELL VARIABLES**

The Helios shell has a number of variables which give it useful information. For example, one of the shell variables is called path, it gives search paths for commands. path tells Helios where to look to find commands and programs. Another shell variable is called prompt, this tells the shell what the command-line prompt should be.

You can display and set shell variables with the set command.
If you issue the set command without any parameters it causes a list of all the variables, together with their current values, to be displayed.

To display a list of shell variables:

```
% set
argv ()
autologout 10
console /IO/window/console
cwd /IO/helios/users/mynname
home /helios/users/mynname
machine /00
path (. /helios/local/bin /helios/bin)
prompt %
shell /helios/bin/shell
status 0
term ansi
user <your login name>
%
```

From this list you can see that the prompt variable is set to '%' and the path variable has two path names assigned to it '/helios/bin' and '.' (the current directory).

**REMMOVING A SHELL VARIABLE**

You can remove a shell variable using the unset command, which takes one parameter:

```
unset <name>
```

where `<name>` is the name of the shell variable you want to remove.
setenv

SETTING ENVIRONMENT VARIABLES
When a shell is started Helios gives it a list of default variables, called environment variables, which give the shell information about the environment in which it is running. The shell copies these variables into its own variable list. The environment variables are also available to any program which is executed from the shell.

The standard environment variables are:

HOME   The user's home directory.
USER   The user's login name.
PATH   The list of directories which are searched when a command is executed.
       (This is not the same as the shell path variable mentioned earlier.
       The shell variable is local to a particular shell whereas the PATH
       environment variable is used to set the default value for the path
       variable every time a new shell is created.)
TERM   The terminal type.
SHELL  The full path name for the shell.

For more information about these variables consult the "The Helios Operating System" manual.

You can change the values of the environment variables using the setenv command, which takes two parameters:

setenv <name> <word-list>

where:

<name> is the name of the environment variable you want to change, or the new variable you want to create.

<word-list> is the string of words to be associated with the variable.
printenv

**LISTING ENVIRONMENT VARIABLES**

You can display a list of environment variables and their values with the `printenv` command.

```bash
% printenv
HOME=/helios/users/myname
SHELL=/helios/bin/shell
USER=<your login name>
PATH=/helios/bin:/helios/users/myname:
TERM=ansi
CONSOLE=/IO/window/console
```

**unsetenv**

**REMOVING AN ENVIRONMENT VARIABLE**

You can remove an environment variable using the `unsetenv` command, which takes one parameter:

```bash
unsetenv <name>
```

where `<name>` is the name of the environment variable you want to remove.

---

source

**SHELL SCRIPTS**

The commands which you type at the shell command line can also be put in a file and executed one by one using the `source` command. Such a file is called a *shell script*. `source` takes as its parameter the name of the file containing the commands and effectively sends each of these commands, one at a time, to the shell interpreter just as if you had typed them yourself.

The advantage of putting a sequence of commands in a command file is that commonly used sequences can be reissued by typing:

```bash
source <file-name>
```

without having to type in each command again.

An example of a shell command file is the `cshrc` file.

**THE LOGIN AND CSHRC SHELL COMMAND FILES**

When you login, Helios looks for two command files in your login directory called `cshrc` and `login`. These files can contain any commands you like. Typically they are used to issue `alias`, `set` and `setenv` commands.

The `login` file is executed once, when you login.
The cshrc file is executed every time a new shell is created. This is normally when you login and every time you issue a shell or wsh command. There is a cshrc file in the guest login directory. It looks like this:

```
set history=20
set savehist=20
alias h history
alias ls ls -F
alias me emacs
set path=( . /helios/local/bin /helios/bin)
```

This file causes the size of the command history record (mentioned earlier in this tutorial) to be set to twenty entries. It also creates three aliases and sets the shell path variable.

You might like to customize the cshrc file and put a copy in your new login directory. For example you may like to add your login directory to the path variable by changing the 'set path...' line to:

```
set path=( . /helios/local/bin /helios/bin /helios/users/myname )
```

---

**ALTERING THE HOST CONFIGURATION**

The default Helios system comes with a file called host.con, in the /helios directory, which contains information about your Helios configuration. If your configuration is different from the default, or if you want to change any of the default settings you will need to edit this file.

The default host.con file looks something like this:

```
host = AT
box = b004
message_limit = 40000
system_image = \helios\lib\nucleus
helios_directory = \helios
bootfile = \helios\lib\nboot.i
floppies = a
Server_windows
```

This file tells Helios that your system consists of a PC/AT compatible host computer and a b004 compatible transputer board.

For information on how to change this file consult "The Helios Operating System" manual.
3.6 USEFUL UTILITIES

Here is a brief list of some of the more useful utilities provided with Helios.

**diff**

EXAMINING THE DIFFERENCES BETWEEN TWO FILES

The `diff` command compares two text files and reports any differences by displaying the lines which differ. This command is particularly useful when keeping track of different versions of the same file. The format of a `diff` command looks like this:

```
diff [-bic] <file-1> <file-2>
```

Where `<file-1>` and `<file-2>` are the files to be compared. The options, `-bic`, are described in “The Helios Operating System” manual.

**fgrep**

SEARCHING FOR A STRING OF CHARACTERS IN A FILE

The `fgrep` command can be used to search a file or group of files for every occurrence of a particular string of characters.

The format of an `fgrep` command looks like this:

```
fgrep [-vnhifuxecls] <string> <file>
```

Where:

`[-vnhifuxecls]` is a list of options. See “The Helios Operating System” manual for more information.

`<string>` is the string of characters to be searched for.

`<file>` is the file or list of files to be searched.

For example, the command

```
fgrep ERROR quote
```

would search the file called `quote` for the word ‘ERROR’. The search is case sensitive unless otherwise specified (by the ‘-i’, ignore case, option).

The command

```
fgrep Helios *
```

would search all the files in the current directory for the word ‘Helios’.

If `fgrep` is successful in finding the string it displays the file name and the line in the file where the string occurs.
The `make` facility provides a convenient method of handling the creation of executable programs from their source code parts. A `makefile` consists of a set of statements that tell the `make` command how to create one or more executable programs. If you look in the `/IO/helios/users/guest/examples/hello` directory you will find a file called `makefile` which looks like this:

```make
# makefile for hello
#

.SUFFIXES:
.SUFFIXES: .o .s .c

.c.o:
   c -c $*.c -o $*.o
   asm -p $*.o -o$*.o

all : hello
hello : hello.c
c hello.c -o hello
```

This file tells `make` how to create the program called `hello`.

If you issue the command

```
make hello
```

(where `hello` is the name of the program you want to make) `make` will look for a file called `makefile`, in the current directory, and look for instructions within that file for making `hello`.

Have a go at making and running the example program called `hello`:

⚠️ You will need a C compiler to do this example.

```
To move to the hello directory:

74 % cd /IO/helios/users/guest/examples/hello
```

DSU///
Advanced Use of Helios

The Helios Getting Started Tutorial

3.7 CONCLUSION

This concludes the Helios Getting Started Tutorial. If you have followed all the examples you should by now have gained sufficient knowledge to start using Helios. In order to keep this tutorial short we have left out a lot of detail about the various commands and their options. Further information is available from the on-line help facility and the documents listed at the end of this book.
FURTHER READING

<table>
<thead>
<tr>
<th>Title</th>
<th>Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MicroEmacs Editor: A Guide for Beginners</td>
<td>H09012</td>
</tr>
<tr>
<td>The Helios Parallel Programming Tutorial</td>
<td></td>
</tr>
<tr>
<td>The Helios Operating System Manual</td>
<td>H09000</td>
</tr>
</tbody>
</table>

SUMMARY OF COMMANDS

This appendix contains a summary of the Helios commands mentioned in this tutorial.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page number</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Background Jobs</td>
<td>60</td>
</tr>
<tr>
<td>alias</td>
<td>Giving a command another name</td>
<td>72</td>
</tr>
<tr>
<td>cat</td>
<td>Concatenating files</td>
<td>53</td>
</tr>
<tr>
<td>cd</td>
<td>Changing Directory</td>
<td>27</td>
</tr>
<tr>
<td>cp</td>
<td>Duplicating files</td>
<td>44</td>
</tr>
<tr>
<td>diff</td>
<td>Examining the differences between two files</td>
<td>85</td>
</tr>
<tr>
<td>emacs</td>
<td>Creating and editing text files</td>
<td>42</td>
</tr>
<tr>
<td>exit</td>
<td>Quitting from a shell</td>
<td>66</td>
</tr>
<tr>
<td>fgrep</td>
<td>Searching for a string of characters in a file</td>
<td>85</td>
</tr>
<tr>
<td>help</td>
<td>Online help information</td>
<td>15</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>history</td>
<td>Displaying a list of recently executed commands</td>
<td>18</td>
</tr>
<tr>
<td>jobs</td>
<td>Listing the Active Background Jobs</td>
<td>62</td>
</tr>
<tr>
<td>kill</td>
<td>Killing a Background Job</td>
<td>62</td>
</tr>
<tr>
<td>login</td>
<td>Logging in</td>
<td>67</td>
</tr>
<tr>
<td>logout</td>
<td>Quitting Helios</td>
<td>49</td>
</tr>
<tr>
<td>ls</td>
<td>Listing the files in a directory</td>
<td>13</td>
</tr>
<tr>
<td>make</td>
<td>Make files</td>
<td>87</td>
</tr>
<tr>
<td>mkdir</td>
<td>Creating new directories</td>
<td>30</td>
</tr>
<tr>
<td>more</td>
<td>Displaying the contents of a file</td>
<td>39</td>
</tr>
<tr>
<td>mv</td>
<td>Moving and renaming files</td>
<td>45</td>
</tr>
<tr>
<td>pr</td>
<td>Printing files</td>
<td>57</td>
</tr>
<tr>
<td>printenv</td>
<td>Listing environment Variables</td>
<td>81</td>
</tr>
<tr>
<td>ps</td>
<td>Displaying a List of Processors and Tasks</td>
<td>64</td>
</tr>
<tr>
<td>pwd</td>
<td>Displaying the Absolute Path Name for the Current Directory</td>
<td>27</td>
</tr>
<tr>
<td>remote</td>
<td>Running a Program on Another Processor</td>
<td>63</td>
</tr>
<tr>
<td>rm</td>
<td>Removing/deleting files and directories</td>
<td>47</td>
</tr>
<tr>
<td>set</td>
<td>Setting the Shell Variables</td>
<td>76</td>
</tr>
<tr>
<td>setenv</td>
<td>Setting Environment Variables</td>
<td>79</td>
</tr>
<tr>
<td>shell</td>
<td>Creating another shell</td>
<td>65</td>
</tr>
<tr>
<td>source</td>
<td>Shell scripts</td>
<td>82</td>
</tr>
<tr>
<td>unalias</td>
<td>Removing an Alias</td>
<td>74</td>
</tr>
<tr>
<td>unset</td>
<td>Removing a Shell Variable</td>
<td>78</td>
</tr>
<tr>
<td>unsetenv</td>
<td>Removing an environment Variable</td>
<td>81</td>
</tr>
<tr>
<td>wsh</td>
<td>Creating another windowed shell</td>
<td>65</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td><strong>Index</strong></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td><code>/printers/default</code>, 57</td>
<td>directories, 21</td>
<td></td>
</tr>
<tr>
<td><code>&lt;</code>, 53</td>
<td>emacs, 42, 68, 70</td>
<td></td>
</tr>
<tr>
<td><code>&gt;</code>, 54</td>
<td>environment variables, 79</td>
<td></td>
</tr>
<tr>
<td><code>&gt;&gt;</code>, 54</td>
<td>exit, 49, 65, 66</td>
<td></td>
</tr>
<tr>
<td><code>&amp;</code>, 60</td>
<td>fgrep, 85, 86</td>
<td></td>
</tr>
<tr>
<td>absolute path name, 25</td>
<td>help, 15, 16...17, 90</td>
<td></td>
</tr>
<tr>
<td>access mode, 14</td>
<td>history, 17, 18, 19...20</td>
<td></td>
</tr>
<tr>
<td>alias, 72, 73, 75, 82</td>
<td>HOME, 79</td>
<td></td>
</tr>
<tr>
<td>alias server, 74</td>
<td>host.con, 84</td>
<td></td>
</tr>
<tr>
<td>background, 60</td>
<td>jobs, 62</td>
<td></td>
</tr>
<tr>
<td>cat, 53, 54...57</td>
<td>kill, 60, 62, 63</td>
<td></td>
</tr>
<tr>
<td>cd, 27</td>
<td>logging in, 66</td>
<td></td>
</tr>
<tr>
<td>command-line prompt, 8</td>
<td>login, 7, 67, 82</td>
<td></td>
</tr>
<tr>
<td>cp, 44, 57</td>
<td>DSL///</td>
<td></td>
</tr>
<tr>
<td>cshrc, 82...83</td>
<td>set, 76, 77...78, 82...83</td>
<td></td>
</tr>
<tr>
<td>current directory, 22</td>
<td>setenv, 79, 82</td>
<td></td>
</tr>
<tr>
<td>diff, 85</td>
<td>shell, 9</td>
<td></td>
</tr>
<tr>
<td>DSL///</td>
<td>SHELL, 79</td>
<td></td>
</tr>
<tr>
<td>Directories, 21</td>
<td>shell, 64, 65, 66...68, 70, 83</td>
<td></td>
</tr>
<tr>
<td>emacs, 42, 68, 70</td>
<td>shell script, 82</td>
<td></td>
</tr>
<tr>
<td>environment variables, 79</td>
<td>source, 82</td>
<td></td>
</tr>
<tr>
<td>exit, 49, 65, 66</td>
<td>standard input, 52</td>
<td></td>
</tr>
<tr>
<td>fgrep, 85, 86</td>
<td>standard output, 52</td>
<td></td>
</tr>
<tr>
<td>help, 15, 16...17, 90</td>
<td>stdin, 52...53</td>
<td></td>
</tr>
<tr>
<td>history, 17, 18, 19...20</td>
<td>stdout, 52, 54...55, 57, 65</td>
<td></td>
</tr>
<tr>
<td>HOME, 79</td>
<td>TERM, 79</td>
<td></td>
</tr>
<tr>
<td>host.con, 84</td>
<td>type of file, 14</td>
<td></td>
</tr>
<tr>
<td>jobs, 62</td>
<td>unset, 78</td>
<td></td>
</tr>
<tr>
<td>kill, 60, 62, 63</td>
<td>unalias, 74</td>
<td></td>
</tr>
<tr>
<td>logging in, 66</td>
<td>unsetenv, 81</td>
<td></td>
</tr>
<tr>
<td>login, 7, 67, 82</td>
<td>USER, 79</td>
<td></td>
</tr>
<tr>
<td>DSL///</td>
<td>wildcards, 33</td>
<td></td>
</tr>
<tr>
<td>set, 76, 77...78, 82...83</td>
<td>wsh, 65...66, 83</td>
<td></td>
</tr>
<tr>
<td>setenv, 79, 82</td>
<td>DSL///</td>
<td></td>
</tr>
</tbody>
</table>