BINS
A new shell for Unix

NAME
BINS stands for « **B**INS **I**s a **N**ew **S**hell ». It's a new command interpreter for Unix

SYNOPSIS
`bins`

COPYRIGHT

DESCRIPTION
BINS is an command language interpreter that executes commands read from the standard input or from a file. BINS also incorporates useful features from Bash. BINS was made with some POSIX specifications and is more or less compatible with it.

INSTALLATION
just untar the bins.tar.gz archive with :
```
tar zxvf bins.tar.gz
```
To compile sources, you have just to go to the *bins* directory created by the archive and type :
```
make
```
since a *Makefile* was generated. Then, all you have to do is to type : `./bins` to run the shell.

UNINSTALLATION
Simply go to the *bins* directory and type :
```
make clean
```
to delete all the compiled files and erase the shell.

OPTIONS
Bins interprets the following options when is invoked :

--history
Equivalent to **-h**. Disables the possibility to handle history in the shell (see **HISTORY**).

--noprofile
Equivalent to **-n**. Avoid to load the configuration files of the shell at startup.

--prompt
Equivalent to **-p**. Permits to modify how the prompt will be displayed in the script (ex: *bins -p test-*).

--version
Equivalent to **-v**. Only displays the version of Bins and exit the shell.

SHELL GRAMMAR
Simple Commands
A simple command is a sequence of optional variable assignments followed by blank-separated words and redirections, and terminated by a control operator. The first word specifies the command to be executed, and is passed as argument zero. The remaining words are passed as arguments to the invoked command.

Pipelines
A pipeline is a sequence of one or more commands separated by the character '|'. The format for a pipeline is:
command | command2
The standard output of *command* is connected via a pipe to the standard input of *command2*.

Compound Commands
A compound command is an instruction only used in shell script to make them more interactive :

goto <*integer*>
Permits to send the script reading to the *integer* line. For example : *goto 3* means to **GO TO** line three. That instruction is from the BASIC language.

if (<*condition*>)
Allow you to test two variables or numbers separated by an operator. Operators accepted are : >, <, ==, !=, >= and <=. If the test is true, the next line will be read. Otherwise, the 2nd next line will be. Ex: *if ($N > 5)*

end
Indicates to the program to finish the script. BEWARE! The *end* instruction must ALWAYS appear in a script to finish it.

Variables

A variable may be assigned to by a statement of the form :
name=[*value*]
Values can be integer or string ou character. In case of a string *value*, all characters are allowed but spaces. To use spaces, you have to place quotes around the *value*. To use variables in any other case that assignment, the '$' sign must be placed before the variable.

Shell Variables

Shell Variables are slightly different from variables since they only can be read. Writing Shell variables is forbidden. Shell variables permit to get information about BINS.
The following variables are set by the shell:

`BINS_VERSION`	The version of BINS
`DATE`	The date in the DD/MM/YY form (example : 28/04/03)
`PWD`	The current working directory
`PROMPT`	The value of this parameter is used as the prompt string.
`TIME`	The time in the HH:MM 24-hour form (example : 19:02)
`USERNAME`	The login name of the current user

REDIRECTIONS

Before a command is executed, its input and output may be redirected using a special notation interpreted by the shell. Redirection may also be used to open and close files for the current shell execution environment. The following redirection operators may precede or appear anywhere within a simple command or may follow a command. Redirections are processed in the order they appear, from left to right.

Redirecting Input

Redirection of input causes the file to be opened for reading. The general format for redirecting input is:
prog < file

Redirecting Output

Redirection of output causes the file to be opened for writing. If the file does not exist it is created; if it does exist it is truncated to zero size. The general format for redirecting output is:
prog > file

Appending Output

Redirection of output in this way causes the file to be opened for appending. If the file does not exist it is created. The general format for appending output is:
prog >> file

ALIASES

Aliases allow a string to be substituted for a word. The shell maintains a list of aliases that may be set and unset with the **link** and **unlink** commands (see **SHELL BUILTIN COMMANDS**). For example : **link ls with ls -al** will permits to execute **ls -al** as soon as **ls** will be typed.
Aliases are created with the **link** commande, listed with the **linklist** command, and removed with the **unlink** command.

JOB CONTROL

The shell associates a job with each process. When Bins starts a job in the background, it prints a line that looks like :

```
[2504] konqueror
```

It indicates that the ID of the process is 2504 and that the name of the program in the background is called *konqueror.*
All the processes in a single pipeline are members of the same job. When a process in the back in the background finishes, Bins prints a line similar to the following one :

```
[2504] konqueror       terminé
```

PROMPTING

When it is ready to read a command, BINS display the prompt variable PROMPT. Bins allows these prompt strings to be customized by inserting some special characters introduced by the slash character. They are decoded as follows:

/a	the year in two numbers (ex:99)
/A	the year in four numbers (ex:1999)
/h	the hours of the current time (ex:11)
/j	the date of day (ex:12)
/J	the name of the day (ex:Mon)
/m	the month in numbers (ex:10)
/M	the name of the month (ex:Oct)
/s	the secondes of the current time (ex:12)
/t	the minutes of the current time (ex:58)
/b	changes the color of the text in blue

/c	changes the color of the text in cyan
/g	changes the color of the text in green
/N	changes the color back to normal
/p	changes the color of the text in pink
/r	changes the color of the text in red
/w	changes the color of the text in white
/y	changes the color of the text in yellow
/D	the full working directory (ex : /usr/local/bin)
/d	the short working directory (ex : bin)
/u	the username
/v	the version of Bins
//	a simple slash

For example, *`prompt It's /J /M, /j and the time is /h:/t:/s>`* will change the prompt to :
`It's Mon Apr, 28 and the time is 14:00:23>`

HISTORY

When the **--***history* option has not been met, the shell provides access to the command history, the list of commands. The shell stores each command in the history list previously typed giving each command a number. To display all the commands available in the history, just type **history**. To recall a specific command, juste type !<n> where *n* is the number given to the latter according to the command **history** (ex : !2). To recall the last command (the one you've just typed), simply type '!!'.

SHELL BUILTIN COMMANDS

commands <*on* | *off* | *status*>
Enable or disable the use of the other builtin commands. If you type `commands off`, the shell commands will be disabled (except **commands**). `commands on` will enable the commands and `commands status` will tell you the status of *commands.*

cd <*directory*>
Change the current directory to *directory.*

del <*string*>
Equivalent to **delete**.

delete <*string*>
Delete the variable which the name is *string*. If you use the keyword **all** instead of the name of the variable (`delete all`), all the variables are deleted.

echo <*string*>
Display the *string* on the standard output. For each word, if it begins with a '$', **echo** understands it as a variable and looks for its value in the variable list. **echo** can also evaluate expressions thanks to the keyword **expr** followed by two numbers separated by an operator. For example : *echo 3 + 1 are expr 3 + 1* will display `3 + 1 are 4`.

exit
Cause the shell to exit. Before it exits, Bins saves the aspect of the prompt and the aliases to load them at the next startup.

history [<*string*>]
Display the list of the last executed commands preceded by a number to recall them (see **HISTORY**). If a *string* is specified, it only displays the history of *string* commands.

link <*word*> with <*string*>
Create an alias between the *word* and the *string*. After created, you'll just have to type the *word* to execute the *string* (see **ALIASES).**

linklist [<*string*>]
Display the values of all the aliases defined by user. If *string* is specified, it only displays the value of the desired alias.

prompt <*string*>
Change the PROMPT variable which displays the prompt. Spaces and special characters are accepted (see **PROMPTING**).

pwd
Display the absolute pathname of the current working directory.

sh <filename>
Execute the shell script named *filename*. It can use special instructions (see **Compound Commands**). Originally the scripts may finish with the '.sh' extension but it isn't a need. Remark lines (starting with #) are allowed.

unlink <*string*>
Delete the alias which the name is *string.*

varlist [<*string*>]
Display the values of all the local variables defined by user. If *string* is specified, it only displays the value of the desired variable.