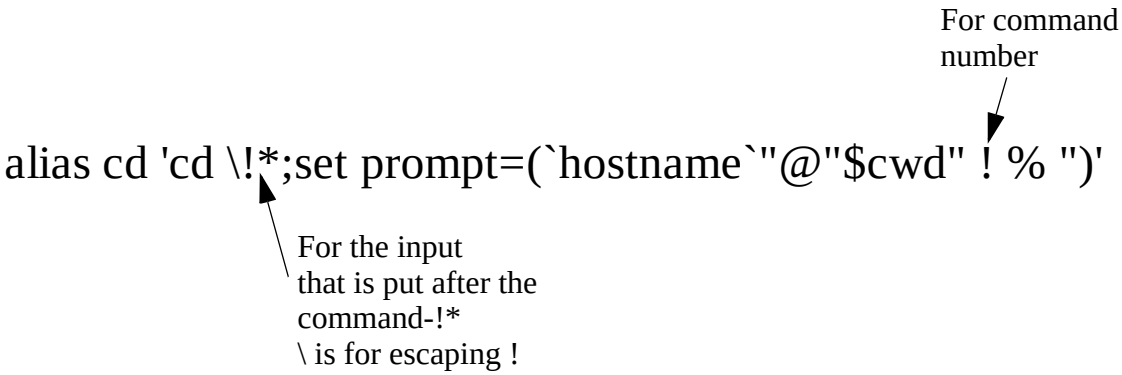
Random Notes on C-shell programming.

1.0 Useful Shell variables:

Knowledge of these variables helps to make a pleasant environment :-) !!. Some of these may be needed in shell scripts also. Typically these are set in the .cshrc file.(For example savehist , history, path etc are usually set in .cshrc)

argv	Array of arguments supplied. $<n> is for nth argument. $* is for all arguments. $#argv has the number of arguments.
noclobber	If set, restricts overwriting of files by redirection. If file needs to be over-written, then redirection symbol " >!" (instead of ">") should be used.
cwd	Variable containing the name of current directory.
ignoreeof	Disables loging out with ctl-D.
nobeep	Disables beeping while file name completion.
nonomatch	Error message are not given when file name doesnt match.
prompt	string to be used as primary prompt.
path	contains the directory names where executables need to be searched. Ex - set path=(~/bin/sparc-sun-solaris2.6/ $path /users/guru/utils) will add other two directories to earliar path.
history	No of commands in history
savehist	No of commands from history to be saves while quiting c-shell.
home	Contains the home directory. ~ refers to this directory. cd < without any arguments> will change the current directory to this directory.
shell	the file where cshell resides.
filec	Enables file name completion with ctl-D and ESC.

2.0 An example :For putting host and directory in prompt.



3.0 First line of the shell script :

is typically /<path>/csh [-bcefinstvVxX] [argument ...]. This is needed as by default all the commands in the scripts are treated to be of bash. In order for interprter to treat the commands to be of c-shell , the path of csh should be given in comments in the first line of shell-script.The leetrs followed by "- " are the options.The functionality of each of the options are tabulated here:

-e	Exit if a command terminates abnormally or yields a nonzero exit status
-f	dont read .cshrc file/ .login file .
-i	Forced interactive. Prompt for command line input, even if the standard input does not appear to be a terminal .
-n	Parse (interpret), but do not execute commands. This option can be used to check syntax etc.
-s	Take commands from the standard input.
-v/V	Echo commands without values of variables substituted.
-x/X	echo commands after all substitutions and just before execution.
-b	Blocks all the further options.

4.0 File inquires:

The boolean information about files can be got by file inquires. A typical use is - if(<file_inquiry>) then....

different file inquiries are tabulated here:

-r filename	True if the user has read access.
-w filename	True if the user has write access.
-x filename	True if the user has execute permission
-e filename	True if filename exists.
-o filename	True if the user owns filename.
-z filename	True if filename is of zero length
-f filename	True if filename is a plain file.
-d filename	True if filename is a directory.

5.0 Arithmatical opeartions :

> + - * / and % have their usual meaning, **But should have a space on either side of the operator .**

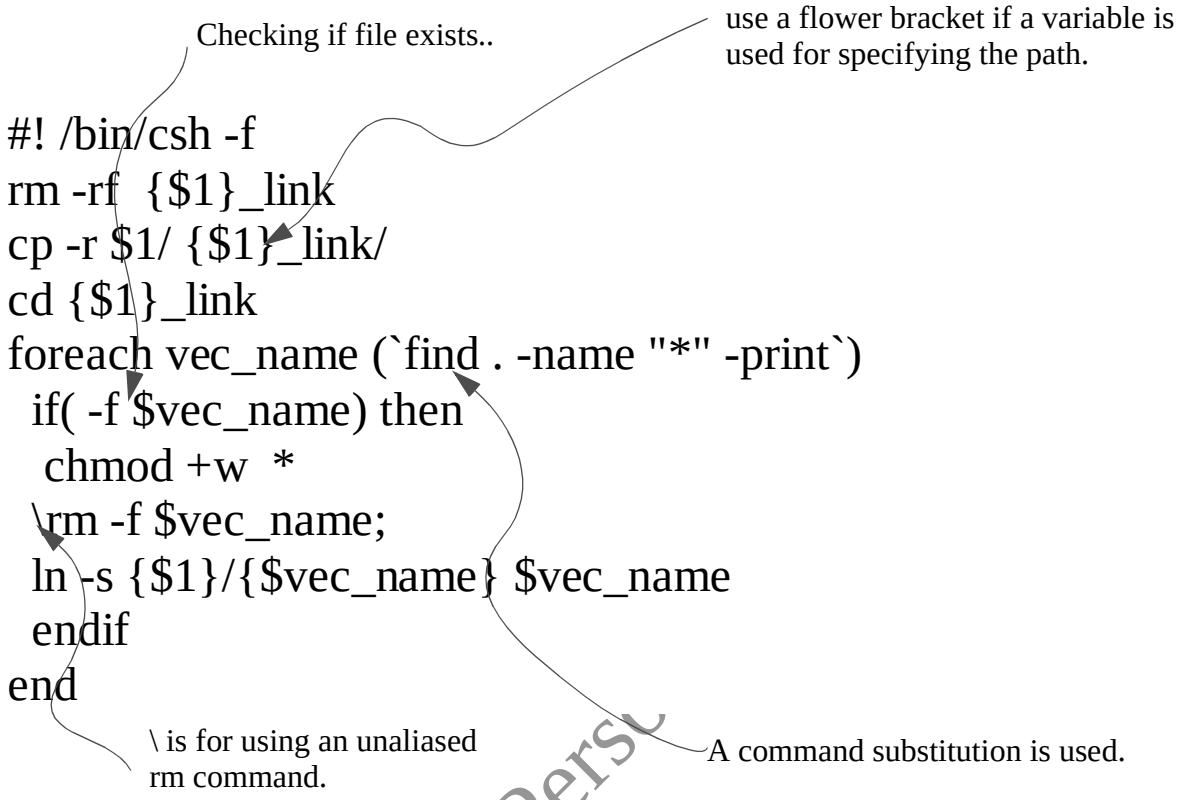
6.0 Logical operators:

For all the operations use **braces appropriately** to avoid any unwanted results!!

>>	<<	!
&	&&	^
|	||	== / !=

7.0 An example :

The following example is for creating a directory with all files linked to those of input dir.



8.0 file name manupulation:

The following table lists different ":" commands and their results.

set dir = /users/guru/sudo_random_data/psdo_data.c	
echo $dir	/users/guru/sudo_random_data/psdo_data.c
echo $dir:h	/users/guru/sudo_random_data
echo $dir:t	psdo_data.c
echo $dir:r	/users/guru/sudo_random_data/psdo_data
echo $dir:e	c
echo $dir:q	/users/guru/sudo_random_data/psdo_data.c
echo $dir:x	/users/guru/sudo_random_data/psdo_data.c

9.0 Syntax of some of the commands:

Set commands:
set var = value
setenv var value

Control statements:
if (expr) command

if(expr) *then*
# commands.
endif

if(expr) *then*
# commands.
else
# commands
endif

foreach var (list)
#commands that can use $var.
#commad *break* will exit the loop
end

while (expr)
#commands
#commad *break* will exit the loop
end

switch (var)
case val1:
#commands
breaksw
case val2:
#commands
breaksw
....
default:
#commands
breaksw

endsw