<u> </u>		Characters Background ich		
<u> </u>	Background job Comment			
•	Home Directory			
	Logical NOT Quote (Strong)			
,	Quote (Strong) Quote (Weak)			
<	Redirect input			
>	Redirect output			
>>	Redirect output + append to file Redirect (pipe) output to next command			
′	_	Separator for pathname directories		
	Separator for shell commands			
[] [}	Start and end a character-set wildcard Start and end a command block			
)	Start and end a subshell			
())	Perform arithmetic			
* ?	Wildcard Wildcard – single character			
<u> </u>	Variable expression			
١		Escape a special character		
n>&m	Desc m	riptor n is a	copy of ouput file descriptor	
n<&m		riptor n is a	copy of input file descriptor	
	m	<u> </u>		
String O				
{varnan		-	Returns word	
{varnan {varnan			Sets and returns word Prints message and exits	
		et:length}	Returns substring	
{varnan	ne:+wo	ord}	If varname is defined,	
Datha		bing one	return word	
attern {varnan		hing opera tern}	Match first from	
. (- arriali	pat		the start	
{varnan	ne##pa	ittern }	Match last from	
{varnan	ne%na	ttern}	the start Match first from	
, annull	ра		the end	
{varnan	ne/pat	tern/replace	e) Match longest	
lyarna	nelle-	ttorn/roals	and replace	
varnan	ne//pa	ttern/replac	ce} Match all and replace	
/ariable	es			
0, \$1, \$2		Positional	parameters	
60, \$1, \$2 6@		"\$1" "\$2" ·	"\$3"	
0, \$1, \$2 @ *		"\$1" "\$2" A string of	parameters "\$3" positional params > 0 f positional params	
0, \$1, \$2 @ * #		"\$1" "\$2" A string of Number of	"\$3" positional params > 0	
60, \$1, \$2 6@ 6* 6# 6? •••••••••••••••••••••••••••••••••	2,	"\$1" "\$2" A string of Number of Exit status	"\$3" positional params > 0 f positional params of last command run	
60, \$1, \$2 6@ 6* 6# 6? •••••••••••••••••••••••••••••••••	2,	"\$1" "\$2" A string of Number of Exit status	"\$3" positional params > 0 f positional params	
60, \$1, \$2 6@ 6* 6# 6? •••••••••••••••••••••••••••••••••	2,	"\$1" "\$2" A string of Number of Exit status function or	"\$3" positional params > 0 f positional params of last command run a myfunction { }	
0, \$1, \$2 @ * # ? unctio	2,	"\$1" "\$2" A string of Number of Exit status function or myfunct	"\$3" positional params > 0 f positional params of last command run	
0, \$1, \$2 6@ * # # ?? unction lefine	ns	"\$1" "\$2" A string of Number of Exit status function or myfunct myfunct	"\$3" positional params > 0 f positional params of last command run n myfunction { }	
io, \$1, \$: @ * # # cunction define all eeyword f / else	ns	"\$1" "\$2" A string of Number of Exit status function or myfunct myfunct local – li tions	"\$3" positional params > 0 f positional params of last command run n myfunction { } tion () { } tion arg1 arg2 mit var scope	
co, \$1, \$2 co	ns	"\$1" "\$2" A string of Number of Exit status function or myfunct myfunct local – li tions	"\$3" positional params > 0 f positional params of last command run n myfunction { } tion () { } tion arg1 arg2 mit var scope	
o, \$1, \$: a the second of th	ns	"\$1" "\$2" A string of Number of Exit status function or myfunct myfunct local – li tions	"\$3" positional params > 0 f positional params of last command run n myfunction { } tion () { } tion arg1 arg2 mit var scope lans, then run y ils, then run y	
o, \$1, \$: a the second of th	ns	"\$1" "\$2" A string of Number of Exit status function or myfunct myfunct local - li tions If x ru If x fa	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope ins, then run y ils, then run y	
io, \$1, \$2 io (a) \$2 io (a) \$1, \$2 io (a) \$2 io (a) \$1, \$2 io (a)	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR N	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope ins, then run y ils, then run y	
50, \$1, \$2 6@ * ## Franction define all teeyword f / else a && y a y a - o y lt, -le, -e ge, -ne	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li itions If x ru If x fa x ANE x OR Integro	"\$3" positional params > 0 f positional params of last command run myfunction { } tion () { } tion arg1 arg2 mit var scope uns, then run y ils, then run y y er comparisons	
50, \$1, \$2 6@ 5* Function define call ceyword f / else c && y c y c -a y	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x ra x ANE x OR - Integr	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope uns, then run y ils, then run y y	
50, \$1, \$5 6@ ** 5:7 Function define (xeyword f / else (xe, xey (x - a y (x -	ns s condi	#\$1" "\$2" A string of Number of Exit status function or myfunct local – li itions If x ru If x fa x ANE x OR - Intege String str1 h str1 h	"\$3" positional params > 0 f positional params of last command run n myfunction { } tion () { } tion arg1 arg2 mit var scope ans, then run y ils, then run y oy y er comparisons tas length > 0 (nonzero) tas length > 0 (zero)	
60, \$1, \$; 6@ * ## 6? iunctio define all teyword f / else 8 & y 1 y 1 - a y 1 - e, -e 1 e, -e 1 e, -e 2 e, -n e 2 i, !=, <, > n str1 d file	ns s condi	#\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANE x OR - Intege str1 h str1 h File e:	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope lins, then run y ils, then run y oy y cer comparisons tas length > 0 (nonzero) tas length 0 (zero) xists and is a directory	
all eyword f / else && y - a y	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR y - Integer str1 h str1 h File e: File e:	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope ans, then run y ils, then run y oy y er comparisons as length > 0 (nonzero) tas length > 0 (zero) xists and is a directory xists	
all seyword f / else & & y & es, es, es fille ef fille ef fille ef fille	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR N - Integer String str1 h File e: File e: File e:	"\$3" positional params > 0 f positional params of last command run in myfunction { } tion () { } tion arg1 arg2 mit var scope lins, then run y ils, then run y oy y cer comparisons tas length > 0 (nonzero) tas length 0 (zero) xists and is a directory	
all eyword f / else & y - o y lt, -le, -e e, -ne e, -ne str1 z str1 d file e file f file r file s file	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANE x OR Integer String str1 h str1 h File e: File e: File e: File e:	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Ins., then run y Ition y Y Y Y Y Y Y Y Y Y Y Y Y Y	
all eyword f / else & y : a	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li itions If x ru If x fa x ANE x OR - Intege String str1 h str1 h File e: File e: User User	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Inst, then run y Ils, then	
so, \$1, \$2 6@ * if contions f else all teyword f else all teyword f else all teyword f else all teyword f f else f f else f file f file f file w file	ns s condi	#\$1" "\$2" A string of Number of Exit status function or myfunct local – li itions If x ru If x fa x ANE x OR - Integr String str1 h str1 h File e: File e: User User User	"\$3" positional params > 0 f positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 Imit var scope Inst, then run y Ist, then run y I	
co, \$1, \$2 compared to the co	ns s condi	#\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANE x OR y - Integr String str1 h str1 h File e: File e: User I User I file, o	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Inst, then run y Ils, then	
all seyword f / else sey e.	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR y - Integer String str1 h str1 h File e: File e: User I User I User I file, o File w read	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Ins, then run y Ils, then	
Variable 50, \$1, \$2 50 \$* 5# 58 Function define call keyword f / else c && y c y c - a y c - le, -e ge, -ne ge, -ne ge, -ne file file r file r file x file W file X file O file O file O file	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR N - Integer String str1 h File e: File e: User I User I User of File w read User of	"\$3" positional params > 0 f positional params of last command run In myfunction { } tion () { } tion arg1 arg2 mit var scope Ins, then run y ils, then run y o y yer comparisons as length > 0 (nonzero) has length 0 (zero) xists and is a directory xists and is a regular file has read permission on file has write permission on file	
so, \$1, \$2 so all ceyword f / else case, -ne e, !=, <, > no str1 d file e file f file r file x file w file N file	ns s condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANL x OR y Integer String str1 h str1 h File e: File e: User I file, o File w read User File w Fi	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Ins, then run y Ils, then	
io, \$1, \$2 io io io io io io io io io i	ns condi	"\$1" "\$2" A string of Number of Exit status function or myfunct local – li tions If x ru If x fa x ANE x OR v - Integr String str1 h str1 h File e: File e: User l User l file, o File w read User c File's user's file1 h	"\$3" positional params > 0 f positional params of last command run In myfunction { } Ition () { } Ition arg1 arg2 mit var scope Ins., then run y Ils., t	

	fi			
or		nit; condition; increment)); do		
	commands; done			
for		r in <i>array</i> ; do		
	comm	*		
	done			
case		xpression in n1) commands ;;		
		n2) commands ;;		
	*) con	nmands ;;		
while	esac while	condition: do		
wille	while condition; do commands;			
	done			
until		ondition; do		
	comm done	unus;		
Useful Commands				
type <cmd< th=""><th>l></th><th>Determine type of command:</th></cmd<>	l>	Determine type of command:		
huiltin con	nd>	-a ; displays all the locations Run builtin commands explicitly		
builtin <cmd></cmd>		` '		
which <cmd></cmd>		Locate the executable of a		
		command: -a ; show all locations		
clear		Clear the terminal screen		
echo "str1	,,	Drint massage to terminal		
ecno "str1		Print message to terminal screen: -e; uses escape sequences like (\n		
		= newline, \t = tab)		
		-n ; supresses automatic newline		
printf <for< th=""><th>mat></th><th>after print Print messages to terminal screen.</th></for<>	mat>	after print Print messages to terminal screen.		
<variables< th=""><th></th><th>Formatting be like:</th></variables<>		Formatting be like:		
		%s – String		
		%-Xs – String wide X chars, left aligned		
		%Xs – String wide X chars, right		
		aligned		
		%d – Integer (%-Xd, %Xd)		
		%f – Float %.Xf - Round to X decimal spaces		
date <opti< th=""><th>ons></th><th>Will display date and time. Formats</th></opti<>	ons>	Will display date and time. Formats		
<+format>		("+%Y-%m-%d"):		
		%Y – Year, %m – month, %d – day, %H – hours, %M –		
		minutes, %S – seconds, (%A		
		uppercase for full name) %a –		
		DayOfTheWeek, (%B) %b - Month Options (-d "yesterday"):		
		"yesterday", "next Monday",		
read <options></options>		Read input from user or file and		
<variable></variable>		store into variable (read var1). Options:		
		-p "Text" : print before input		
		-a : store the input in array		
history <options></options>		Display the command history for that session. Options:		
- CPCIOIIS		-c : clear the history		
		-X : print the last X commands		
		-a : appends history to bash history file		
		-d X : deletes the command with		
		index X from history		
sleep	0>	Delay the execution of a script.		
<num_time></num_time>		Num_time: Xs : delay for X second(s) (default)		
		Xm : delay for X minute(s)		
		Xh : delay for X hour(s)		
man		Opens the manual pages for the		
<pre><command/></pre>		<command/> .		
Is <option< th=""><th>s></th><th>List the files and directories in the</th></option<>	s>	List the files and directories in the		
<path></path>		current working directory or given path. Options:		
		-l : list detailed view for files		
		-a : show all files, even hidden		
nu-d		-alp: ???		
pwd		Display the current working directory.		
cd <directory></directory>		Change the current working		
		directory. <directory>:</directory>		
		'/path' : changes directory to path		

commands; fi

	'': changes to parent directory of
	the current one
	'~username' : changes to home directory for username
	'-' : changes to previous working
mkdir	directory used Creates new directory. <directory></directory>
<directory></directory>	can be:
	'd1' : creates new directory called d1
	'd1' 'd2' 'd3' : creates more
	directories in the current one
	-p 'd1/d2' : creates d1 and another directory d2 as d1's child
rmdir	Works the same as mkdir, but it
<directory> cat <file></file></directory>	deletes the directory if it is empty. Display the contents of the file on
cut sines	the terminal. <file>:</file>
	'file.txt' : displays file.txt 'f1.txt' 'f2.txt' : displays files
	consecutively
	-n 'file.txt' : displays file.txt with
more, less, od,	numbered lines More and less are both text
hexdump	viewers, od gives octal output and
cp <source/>	hexdump hexadecimal. Copy files or directories from
<destination></destination>	source to destination.
	cp file /path : copy file to path cp -r directory /path : copy
	directory with all its contents to
mv <source/>	path Moves files or directories from
<pre><destination></destination></pre>	source to destination.
	mv file /path : move file to path
	mv directory /path : move directory to path
	mv file.txt newfile.txt : renames
uniq <options></options>	file.txt to newfile.txt Removes all consecutive lines.
<file></file>	Options:
	-c : also counts the amount of duplicates
	-i : ignores the case
	-d : outputs only duplicates -u : outputs only the unique
rev <file></file>	Reverse the characters in each line
tr <options></options>	of the input stream or file Translate or delete characters.
<set1> <set2></set2></set1>	Set1 is translated to Set2.
<file></file>	-d : removes the characters -c : complement the Set1
wc <options></options>	Counts the number of lines, words,
<file></file>	bytes. Options: -I : only counts the lines
	-w : only counts the words
	-c : only counts the bytes
grep <options> <pattern> <file></file></pattern></options>	Search for specific pattern or regular expression. Options:
	-i : ignore case
	-v: invert the match (print only the lines not matching the pattern)
	-w : match only whole words
	-n : print the line numbers for each match
	-r : search recursively through
shift <x></x>	directories Shift the positional parameters to
	the left. X is number of positions to
jobs <options></options>	shift. Display a list of jobs that are
Jobs roptions	currently running in the
	background or are suspendedI: also displays PID of a job
	-p : displays only the PIDs
	-r : displays the running jobs -s : displays the stopped jobs
fg <jid></jid>	Bring a job that is running in the
ha all De	background to the foreground.
bg <jid></jid>	Start a suspended job in the background.
disown	Remove jobs from shell's job
% <jid></jid>	control. (disown %2 : removes job with JID 2)
ulimit <options></options>	Display the resource limits of the
	current shell and its childrena : displas all current limits