Control Structures:

* If… then

If… then… else, if… then… elif

* For… in

For index in argument list, do something, until done

For index do something, until done

* While

While this is true, do something, until done

* Until

Similar to while-loop. Until this is true, do something until done.

Break and continue, use with loops to break out of loop and continue loop

* Case

Basically, a switch statement.

* Select

Displays a menu. Select varname in [ argument list ]

Here document:

Redirect input from a shell script from within the shell script

Available here instead of there

$ Cat > birthday

Grep -I “$1” <<+

Zach January 23

May June 14

Nancy June 22

$ ./birthday Zach

Zach January 23

$ ./birthday June

May June 14

Nancy June 22

File Descriptors:

Number associated to the file – standard input, output, error -> 0, 1, 2

Opening – exec n> outfile, exec m< infile

Duplicating – exec n<&m (input) exec n>&m (output)

Closing – exec n<&-

Test -t tests if file descriptor is associated with terminal, 1 for true, 0 for false. Tests is standard input or output or error is coming from or going to terminal

Parameters:

Positional – command name and command line arguments, value cannot be changed by an assignment statement

$0 – name of calling program

$1-$n – positional parameters, 1 is the first argument, 2 is the second … n is the nth

If n > 9, must enclose n with {}.

To initializes use set keyword – set this this2 this3 == $1(this), $2(this1), $3(this2)

Shift promotes parameters – discards the previous args and replaces it with next, there is no unshift so cannot be undiscarded

$\* and $@ - expands to all positional parameters. Both are the same until “$\*” then yields one argument with space between the parameters, “$@” then yields a list of arguments where each argument is a parameter.

$# - number of parameters

$$ - PID number

$! – PID number of most recent background process

$? – Exit status, 0 for true, 1 for false

$- - Flags of options that are set

$\_ - last argument of previously executed command

Variables:

Shell variables – available only in the shell it was created in

Environment variables – global variables/exported variables live in the shell’s environment

Child processes inherit environment from parent, so all environment variables are inherited

Process locality – variables are local to a process.

Export – puts variables in the environment

Printenv – displays variable names and values

Env – runs a program in a modified environment, displays a list of environment variables when used without command line

:- uses a default value, and := assigns a default value

Array variables – array = ([0],[1],[2],…,[n]) array[0] = ${array[0]}

Functions run in the same environment as the shell so variables are implicitly shared by a shell when a function calls it

Builtin Commands:

Type, read, exec, trap, kill, eval, getopts (parses options) – kinda self explanatory in their names

Good practice = specifying the shell, verifying number and type of arguments, displaying standard error messages, etc.