```
##makefile stuff##

target:   dep1 dep2 dep3 ...     # Dependency line
            cmd                  # Command line

example: called with make -f makefile_rander

makefile_rander

rander: rander.o twister.o
               g++ -o  rander rander.o twister.o
rander.o: rander.cpp rander.h
              g++ -g -c rander.cpp
twister.o: twister.cpp twister.h
              g++ -g -c twister.cpp

makefile: called with make

CC           = g++
CFLAGS       = -g
TARGET       = rander
OBJS         = rander.o twister.o linear.o
$(TARGET): $(OBJS)
               $(CC) -o $(TARGET) $(OBJ)
rander.o: rander.cpp rander.h
              $(CC) $(CFLAGS) -c rander.cpp
twister.o: twister.cpp twister.h
              $(CC) $(CFLAGS) -c twister.cpp
linear.o: linear.cpp linear.h
              $(CC) $(CFLAGS) -c linear.cpp
clean:
              /bin/rm -f *.o $(TARGET)

#!/bin/bash

echo "just parens"
var="Hello"
echo $var
(var="Hi"; echo $var); echo $var

echo
echo "with export"
var="hi"
export var
echo $var
(echo $var; var="Hello"; echo $var); echo $var

echo
echo "curly braces"
var="Hi"
```

```
export var
echo $var
{ echo $var; var="Hello"; echo $var;}; echo $var

just parens
Hello
Hi
Hello

with export
hi
hi
Hello
hi

curly braces
Hi
Hi
Hello
Hello

##general bash##

$0 = command/script
$# = number of args including script name
$* = list all args except for script name
$? = last exit status

##switch case##

case $str in
       one)
       echo 1;;
       two)
       echo 2 ;;
       five)
       echo 5;;
esac

if [[ $str == "one" ]]; then
       echo 1
elif [[ $str =="two" ]]; then
       echo 2
else
       echo 0
fi

##file path stuff##

if [[ -d $PASSED ]]; then
```

```
    echo "$PASSED is a directory"
elif [[ -f $PASSED ]]; then
    echo "$PASSED is a file"
else
    echo "$PASSED is not valid"
    exit 1
fi

##for loop##

for file check

for f in *; do
  echo "File -> $f"
done

##math stuff##

# getting the number into the $1 slot
shift
#checking number is in the right range
if ! [[ $1 =~ ^[0-9]+$ ]] || [[ $1 -gt 100 ]] || [[ $1 -lt 10 ]];
then
    usage
    exit 1
fi

#initializing things for while loop
lowestNumber=$(($1 / 2))
currentNumber=${lowestNumber}
moddedNumber=$(($1 % $lowestNumber))
while [[ "$currentNumber" -gt 1 ]]
do
    # checking for perfect divisibility for every number
    moddedNumber=$(($1 % $currentNumber))
    if [[ ${moddedNumber} = 0 ]]; then
        lowestNumber=${currentNumber}
    fi
    currentNumber=$(( $currentNumber - 1 ))
done

#      The name of your script;
echo $0

#      Current date and time;
date
#      The name of the user;
whoami
#      The name of current working directory;
pwd
```

```
#      The name of UNIX machine;
hostname
#      The name of login shell;
echo $SHELL

# checks that the file exists
if [ -f "$file" ]; then
    #  Contents of the required file;
    echo "Contents of $file {"
    cat ${file}
    echo "}"
    #  Number of text lines in the file;
    echo "Number of lines in $file"
    wc -l ${file}
else
    echo "$file doesn't exist"
fi
#      Listing of the required directory;
# checks that path exists
if [ -p "$path" ]; then
    echo "Listing of $path {"
    ls $path
    echo "}"
else
    echo "Path does not exist"
fi
#      Total number of parameters of the script;
echo $#
#      Calendar for October of the current year;
cal -m 10
#      Disk usage;
df -h
#      Current number of users in the system;
w | wc -l
#      Current time.
date +%r

# here file
cat << 'EOF' > myContacts
Phillip
phill

notme
asdf
EOF

# null out EmptyDir.txt
truncate -s 0 EmptyDir.txt

##check dir if empty##
```

```
checkDir() {
  directory=$1
  notEmpty=false
  # if anything exists in the current directory
  if [[ "$(ls -A $directory)" ]]; then
    echo Files in subdirectors of $1
    #  for every sub file or directory in the current direcotry
    for sub in $directory/*; do
      # if the current path is a file then output the name
      if [[ -f $sub ]]; then
        echo $sub
        # mark the file as not empty
        notEmpty=true
      # else call the checkDir function again
      else
        checkDir $sub
        # if the current path is empty put it at the bottom of empty
dir
        if [[ "$notEmpty"=false ]]; then
          echo $sub >> EmptyDir.txt
        fi
      fi
    done
  fi
}

#read file
while read -r contact ; do
  if echo $contact | grep "$regex" ; then
    matched="true"
  fi
done < myContacts

##sed##

from a file named Cars
this will replace all upper case letters with lower case letters
replace all commas with a tab
put the results into Result.txt

sed 's/[A-Z]/[a-z]/g' Cars | sed 's/,/\t/g' > Results.txt

$ = end of line
^ = start of line
[^...] not containing
[...] containing

##other##
```

```
extern int a;
```

a doesn't need to be initialized in its main file, the compiler will expect it to be declared in a header or something

malloc

```
int *arr;
arr = malloc ( <size> * sizeof(int));
```

malloc for struct array

```
Structtype **data;
data = (Struct**) malloc(sizeof(struct Structtype) * 5);
```

```
##getopt##
//loop over options
while(( o = getopt (argc, argv, "i:o:h::")) != -1) {
    switch (o) {
        case 'h':
            printUsage();
            return 0;

        case 'i':
            usedI = true;
            strncpy(inputName, optarg, 100);
            break;

        case 'o':
            usedO = true;
            strncpy(outputName, optarg, 100);
            break;

        case '?':
            return 1;
    }
}
```

\n newline
\t tab
\b backspace

atof(str) tring to double
strtod(str, ptr) string to double ptr
atol string to int
strol(str, ptr, base) string to int of base 0 to 36
scanf(“%d”,&x);   reads an integer
scanf(“%f”,&f);   reads a float
scanf(“%s”,s);   reads a string, NO &
strcat(s1,s2) Appends a copy of string s2 to end of string s1

strncat(s1,s2,n) Appends at most n characters from s2 to s1
strcpy(s1,s2) Copies s2 to s1 until it reaches NULL character
strncpy(s1,s2,n) Copies s2 to s1 until NULL or after n characters
strdup(s) Duplicates string and returns pointer to new str
strlen(s) Returns number of characters in s(non-null)
strcmp(s1,s2) Compare strings s1 and s2
strncmp(s1,s2,n) Compare first n characters of s1 and s2
strcasecmp(s1,s2) Same as strcmp but ignore case of characters
strncasecmp(s1,s2,n) Same as strncmp but ignore case of characters
strchr(s,c) Returns pointer to first occ. of character c in s
strrchr(s,c) Returns pointer to last occ. of character c in s
strpbrk(s1,s2) Returns a pointer in s1 to first occ. of any character in s2
strspn(s1,s2) Returns length of s1 that is entirely composed of chars from s2
strcspn(s1,s2) Returns length of s1 that is not composed of chars from s2
strstr(s1,s2) Returns pointer to location of substr s2 in s1
strtok(s1,s2) Used to break up string s1 by a token in string s2
memcpy(s1,s2,n) Copies n characters from s2 to s1
memccpy(s1,s2,c,n) Copies characters from s2 to s1 until char c or max n
memchr(s1,c,n) Finds c in s1 and returns location (only check max of n)
memcmp(s1,s2,n) Compare strings up to n characters
memset(s1,c,n) Sets first n characters in s1 to the character c

```
//memory allocation//
char *c;
int SIZE = 100;
c = malloc( SIZE * sizeof( char ) )

complicated_type_t *ptr;
ptr = ( complicated_type_t *) malloc( sizeof( complicated_type_t) );

free(ptr);

//multi dimensional allocation//
int **brd = ( int ** ) malloc( sizeof( int * ) * rows);
for (int i = 0; i < cols; i++) {
   brd[i] = ( int * ) malloc( sizeof(int) * cols);
}

for (int i = 0; i < cols; i++) {
   free(brd[ i ]);
}
free(brd);

realloc( ptr,  new_size);
```

```
calloc( int num, int size_in_bytes);
Equivalent to malloc, just different format
a = (int *) calloc( elements, sizeof( int ));

//processes//

pid is an int that id's a process
pid 0 swapper
pid 1 is parent of all processes
pid 2 pagedaemon handles paging

exec() used to run
called with

All these arguments are of type char *, including null

path must be the name of an executable program file

Example of calling ls -l *.txt
int execl( path, arg0, arg1, ..., argn, null);
execl("/bin/ls", "ls", "-l", "*.txt", NULL);
returns -1 if unsuccessful
int pid = fork();
Parent receives PID of child
Child's PID and PPID are different
Use ^Z to suspend a running process
getpid() function lets a process get its own PID
void exit( int status);
Terminates process that issued it

Rightmost byte of status returned as a status code
All open file descriptors are closed
All standard I/O streams closed and buffers flushed
wait system call sleeps process until at least one child returns
Called with
int pid = wait ( int *returnstatus);
pid is pid of child process that ended
returnstatus is given status of exit of child process

waitpid(int pid, int *status, int options)
Waits for a particular pid or range of pids

char * fgets ( char * str, int size, FILE * stream );
FILE * stream = fopen ("file name", r,w,a) read, write, append
r+ read or write
w+ is write but can also read
a+can append or create new file and can also read

int fputs( char *str, FILE *fd);
```

```
//Writing to a binary file//
fread( char * buf_ptr, int size, int nitems, FILE * fd);
fwrite( char * buf_ptr, int size, int nitems, FILE * fd);
Both transfer a specified number of bytes beginning at a location in
memory to a file at the location specified by fd
buf_ptr is a pointer to a buffer
size is the number of bytes in one element of the buffer
nitems is number of elements in the buffer

int i = 19;
FILE * fd;
fd = fopen("binarysequence.dat","w" );
fwrite( &i, sizeof( int ), 1, fd);

fprintf( fd, "%d", number );
fwrite( &number, sizeof( int ), 1, fd );

fflush(FILE * stream) to flush the buffer
```