

1 workflow

```
1 #-----#
2 # TMUX-SHELL #
3 #-----#
4
5 $ C-l # clear screen
6 $ C-w # delete word
7 $ C-_ # undo
8 $ C-c # kill
9 $ C-d # exit
10 $ C-Z # suspend process
11 $ fg # restore process
12 $ C-a # jump to the strt of the line
13 $ C-e # jump to the end of the line
14 $ open <directory path> # open in finder
15 #-----#
16 $ C-space "" # split pane
17 $ C-space % # split pane
18 $ C-space arrow # jump panw
19 $ C-space { # move pane
20 % C-space } # move pane
21 $ C-space x # kill pane
22 $ C-space q # show pane number
23 $ C-space q 1 # goto pane 1
24
25 $ :resize-pane -D # resizes down
26 $ :resize-pane -U # resizes upward
27 $ :resize-pane -L # resizes left
28 $ :resize-pane -R # resizes right
29 $ :resize-pane -D 10 # resizes down by 10 cells
30 $ :resize-pane -U 10 # resizes upward by 10 cells
31 $ :resize-pane -L 10 # resizes left by 10 cells
32 $ :resize-pane -R 10 # resizes right by 10 cells
33 #-----#
34 $ C-space s # list session
35 $ C-space :new # new session
36 $ tmux kill-session -t <name> # kill session
37 $ tmux attach -t <name> # re-attach session
38 #-----#
39 $ ssh hostname # hostname-c_user SSH port22
40 $ ssh -i foo.pem hostname # hostname-identity file
41 $ ssh user@hostname # hostname-user-SSH port22
42 $ ssh user@hostname -p 8765 # hostname-user-custom port
43 $ ssh ssh://user@hostname:8765 # hostname-user-custom port
44 $ scp .txt ubuntu@hostname:/home# copy foo.txt into remote dir
45 #-----#
46 $ cat foo.c # create file with content
47 $ touch foo.c # create file without content
48 #-----#
49 $ mkdir test # create dir
50 $ rmdir test # remove dirgit
51
52 $ cd ../snippets/ # navigate subdir of parnt dir
53 $ cd ../mmio.h # navigate curr dir
54
55 $ cp ./file.xyz ../target/ # copy into subdir of parent
56 $ mv Makefile Makefile_ex # rename old->new
57 $ mv * ../ # move all upper folder
58
59 $ && # chain command in bash
```

```
1 #-----#
2 # MAKE #
3 #-----#
4
5 # compiling with linking in non-default name '-o'
6 # read.o is dependency
7 # if timestap changed on read.o it will be re-linked
8 read: read.o mmio.o
9 cc -fopenmp -O4 -Wall -g read.o mmio.o -o read
10
11 # compiling without linking '-c';
12 # multiple pre-requisites used if anything changed
13 # -Wall gives all the warning; -g turns on the debugger
14 read.o: example_read.c ../lib/mmio.c
15 cc -fopenmp -O4 -c -Wall -g example_read.c -o read.o
16 cc -fopenmp -O4 -c -Wall -g ../lib/mmio.c -o mmio.o
17
18 clean:
19 rm -f read read.o mmio.o
```

```
1 # 1_login remotely
2 $ ssh -X sid@crescent.central.cranfield.ac.uk
3 $ password
4 $ module load fosscuda/2019b
5 $ export CC=$(which gcc)
6
7 # 2_create source file
```

```
8 $ vim ex1.c
9 $ vim Makefile
10
11 # 3_compile manually / with Make / recompile with Make
12 # o gives it a custom name instead of default
13 $ gcc -fopenmp -O4 -o ex1 ex1.c
14 $ make ex1
15 cc -Wall -g ex1.c -o ex1
16 $ make clean
17 rm -f ex1
18 $ make ex1
19 cc -Wall -g ex1.c -o ex1
20
21 # 4_run executable
22 $ ./ex1
23 # or add input data and run
24 $ ./read ../test/cage4.mtx
25
26 # 5_create, submit job file
27 $ vim ex1.sub
28 $ qsub ex1.sub
29
30 # 6_status
31 $ qstat
32 $ ls
33 $ more openMP.02300565
34
35 # 7_copy remotely into local
36 $ scp sid@crescent.central.cranfield.ac.uk:
37 openMP.o230565 /Documents/lib/ex2_3.test
```

```
1 #-----#
2 # GIT #
3 #-----#
4
5 # create a repo on github
6 # then create a local project folder
7 $ mkdir SpMV_OpenMP
8
9 # initialise git on current folder and push it
10 $ git init
11 $ git add README.md
12 $ git commit -m "first commit"
13 $ git branch -M main
14 $ git remote add origin git@github.com:marcellgyorei/
15 spmv_openmp.git
16 $ git push -u origin main
17
18 # or clone repo
19 $ git clone git@github.com:marcellgyorei/SpMV_OpenMP.git
20
21 # check changes have been made before committing
22 $ git status
23 # what changes have been made
24 $ git diff
25 # see changes on particular file
26 # which lines have been added/deleted
27 git diff R/modified.R
28
29 # use one global .gitignore whenever check git status
30 $ nvim ~/.gitignore_global
31 # add lines into it
32 *-
33 *-
34 .DS_Store
35 .Rhistory
36 .RData
37 $ git config --global core.excludesfile ~/.gitignore_global
38
39 # check log of commits
40 $ git log
41 # compressed log
42 $ git log --pretty=oneline
43 # commits of certain author
44 $ git log --author=marcellgyorei
45 # only files have changed
46 git log --name-status
47 # tree log
48 $ git log --graph --oneline --decorate --all
49
50 # drop local changes-commits, fetch latest history from server
51 $ git fetch origin
52 $ git reset --hard origin/main
53
54 # delete local git repo
55 $ rm -fr .git
56 # verify status
57 $ git status
58
59 # delete local folder and re-clone it
60 $ rm -rf -/spmv_openmp
```

```

61 $ git clone git@github.com:myname/myproject.git ~/spmv_openmp
62
63 # add a folder content
64 $ git add foldername/\*
65
66 $ git add --all
67 $ git commit -am "<commit message>"
68 $ git push
69
70 $ git pull --rebase
71
72 # is there are unstaged changes list files that prevent pull
73 $ git status
74 $ git restore .DS_Store

```

```

1  /*-----*/
2  /* VIM_MODE */
3  /*-----*/
4
5  save as ex1                :w! ex1
6  quit/save & quit          :!q    :wq
7  insert/command mode       i      ESC
8
9  /*-----*/
10 /* VIM_FORMAT */
11 /*-----*/
12
13 indent line forward/backward i C-t i C-d
14
15 /*-----*/
16 /* VIM_SELECT-COPY-PASTE */
17 /*-----*/
18
19 line selection              V
20 select word forward/backward vw vb
21 /*-----*/
22 copy lines by number        :<number>yy
23 copy current line           yy
24 copy selection              y
25 /*-----*/
26 paste buffer before/after crsr p P
27 undo                        u
28
29 /*-----*/
30 /* VIM_REPLACE */
31 /*-----*/
32
33 replace text                 :%s/<match>/<replace>
34 replace with '               r'
35 switch case under the char  ~
36
37 /*-----*/
38 /* VIM_SEARCH */
39 /*-----*/
40
41 show lines match             [I
42 /*-----*/
43 search forward/backward      /<match> ?<match>
44 search word nrst frwd/bckwrd * #
45 repeat search forward/backward n N
46
47 /*-----*/
48 /* VIM_JUMP */
49 /*-----*/
50
51 next/prev page               C-f C-b
52 half page up/down           C-u C-d
53 /*-----*/
54 top/middle/bottom line      H M L
55 set line numbering           :set number
56 goto line                   :<line number>
57 /*-----*/
58 to first/last line of a text gg G
59 /*-----*/
60 end of the line              $
61 first char of the line [blank] 0
62 first char of the line       ^
63 /*-----*/
64 next word                    w W
65 end of the word              e E
66 prev word                    b B
67 prev space                   F[]
68 /*-----*/
69 next 'e' char in line        fe
70 repeat [opposite]            ; ,
71 /*-----*/
72 bracket to bracket           %
73 left/right/down/up          h l j k
74
75 /*-----*/
76 /* VIM_DELETE */

```

```

77 /*-----*/
78
79 until first/last line in text dgg dG
80 bracket content              dt%
81 /*-----*/
82 current line                  dd cc
83 current & prev/next line     dk dj
84 until end of the line         d$
85 /*-----*/
86 start of the word forward     dw dW cw
87 end of the word forward       de dE
88 start of the word backward    db dB
89 /*-----*/
90 until " char                  dt"
91 current char                  x

```

2 c

```

1  /*-----*/
2  /* USER DEFINED FUNCTION EXAMPLE */
3  /*-----*/
4
5  // pre-processor directive necessary when using math library
6  #include <math.h>
7
8  // function prototype
9  double gen_sqrt(double);
10
11 // main function
12 int main()
13 {
14     // variables
15     double val,sqroot;
16
17     // ask the user to enter a real number
18     printf("Enter a floating point value > 0");
19
20     // get the value from the user
21     scanf("%lf",&val);
22
23     // call the function to compute the generalised sq root
24     sqroot=gen_sqrt(val);
25
26     // print out the result
27     printf("The generalised square root of %lf is %lf\n",val,
28           sqroot);
29
30     return 0;
31 }
32
33 // user-defined function gen_sqrt
34 double gen_sqrt(double x)
35 {
36     double result;
37     if(x <0.0)
38     {
39         result=-sqrt(-x);
40     }
41     else
42     {
43         result=sqrt(x);
44     }
45     return (result);
46 }

```

```

1  /*-----*/
2  /* VARIABLES */
3  /*-----*/
4
5  auto      break      char      double
6  else      extern     int      return
7  struct    case       enum      long
8  register  switch     typedef   union
9  const     continue   float     for
10 short     unsigned    default   goto
11 signed    sizeof     void      do
12 static    volatile   if        while

```

```

1  /*-----*/
2  /* DATA TYPES */
3  /*-----*/
4
5  Type      PC  Dec MIPS  Dec Alpha  Dec Alpha
6             (OSF/i)  (ULTRIX)  (OPEN VMS)
7
8  char       1   1         1           1
9  short int  2   2         2           2
10 int        2   4         4           4
11 long int   4   4         8           4
12 float      4   4         8           4
13 double     8   8         8           8

```

```

1  /*-----*/
2  /* INCREMENT */
3  /*-----*/
4
5  // output i: 1
6  int main()
7  {
8      int i=0;
9      printf("i: %d\n",++i);
10     return 0;
11 }
12
13 // output i: 0
14 int main()

```

```

15 {
16     int i=0;
17     printf("i: %d\n",i++);
18     return 0;
19 }

```

```

1  /*-----*/
2  /* LOOP */
3  /*-----*/
4
5  /*
6  [expression-1]: evaluated before the first loop iteration
7  [expression-2]: determines whether to terminate the loop;
8                  evaluated before each loop iteration
9  [expression-3]: evaluated after each iteration
10 */
11
12 #include <stdio.h>
13
14 void action1();
15 void action2();
16
17 int main()
18 {
19     int a;
20
21     for(;;)
22     {
23         printf("Enter a choice\n");
24         printf("\t 1. Action 1\n");
25         printf("\t 2. Action 2\n");
26         printf("\t 3. Exit\n");
27
28         scanf("%d",&a);
29
30         switch(a)
31         {
32             case 1: action1();
33                     break;
34             case 2: action2();
35                     break;
36             case 3: printf("Exit...\n");
37                     break;
38             default: printf("Incorrect choice\n");
39         }
40     }
41     return 0;
42 }
43
44 // action routines
45 void action1()
46 {
47     printf("This is the action1 routine\n");
48 }
49 void action2()
50 {
51     printf("This is the action2 routine\n");
52 }

```

```

1  /*-----*/
2  /* JUMP STATEMENTS */
3  /*-----*/
4
5  // never use goto unless for error handling
6
7  for (...)
8  {
9      ...
10     for (...)
11     {
12         ...
13         if (disaster)
14             goto error;
15     }
16     ...
17 }
18
19 error:
20     /* error handling */
21     return;

```

```

1  /*-----*/
2  /* FUNCTION PROTOTYPES */
3  /*-----*/
4
5  // function definition
6  char func(int lower, int *upper, char (*func)(), double y )
7  {}
8
9
10 // prototype declaration v1
11 char func(int lower, int *upper, char (*func)(), double y);

```

```

12 // v2
13 char func(int a, int *b, char (*c)(), double d);
14
15 // v3
16 char func(int, int *, char (*)(), double);
17
18

```

```

1 /*-----*/
2 /* DYNAMIC MEMORY */
3 /*-----*/
4
5 pointer = malloc(number-of-bytes);
6
7 // simple.c

```

```

1 /*-----*/
2 /* BUFFERED I/O - PRINTF & FPRINTF */
3 /*-----*/
4
5 printf(format-string, argument, ...)
6
7 printf("%10.2f\n", i);
8 // %10.2f: field specification
9 // m[10]: minimum field width
10 // p[2]: precision; number of digits after the decimal point
11 // f: conversion character
12 // displays a floating-point number in "fixed decimal"
13
14 // conversion characters:
15 %d - prints in short int
16 %c - prints integer as character
17 %o - prints in octal
18 %x - prints in hexadecimal
19 %f - prints both float and double
20 %l - prints in long int
21
22 // examples:
23 // print a floating point number with 2 dig after dec point
24 printf("Profit: %.2f\n", profit);
25 profit: $2150.48
26 // print the number use at least 3 characters
27 printf("Number: ->%3d<-\\n", 12);
28 ->.12<-
29 // print with at least 3 characters; left-justify it
30 printf("Number: ->%-3d<-\\n", 12);
31 ->12.<-
32 // print with at least 3 characters
33 printf("Number: ->%3d<-\\n", 1234);
34 ->1234<-
35
36 // predefined files:
37 stdin - standard in; normal program input
38 stdout - standard out; normal program output
39 stderr - standard error; error output
40
41 // printf replaces fprintf(stdout, ...)
42 // writing to a predefined file and/or opened file:
43 fprintf(stdout, "Everything is OK\\n");
44 fprintf(stderr, "ERROR: Something bad happened\\n");

```

```

1 /*-----*/
2 /* BUFFERED I/O - FGETS & SSCANF */
3 /*-----*/
4
5 // reading data from opened file and/or predef files
6 fgets(line, sizeof(line), stdin);
7 sscanf(line, "%d %d", &aInteger, &anotherInteger);
8
9 // general form fgets:
10 char* result = fgets(buffer, size, file);
11
12 // result: is a pointer to the string that was just read
13 // (buffer) or NULL if end of the file has been reached
14
15 // buffer: is a chrctr array where the line is to be placed
16
17 // file: is a file handle indicating which file to read
18 // (stdin in this case)
19
20 if (fgets(line, sizeof(line), stdin) == NULL)
21 {
22     fprintf(stderr, "ERROR: Expected two integers, got EOF\\n");
23     return (ERROR);
24 }
25
26 // ampersands used because it needs to modify the arguments
27 // therefore arguments must be passed by address
28 // sscanf returns the number of items it converted
29 if (sscanf(line, "%d %d", &aInteger, &anotherInteger) != 2)

```

```

29 {
30     fprintf(stderr, "ERROR: Expected two integers.\\n");
31     return (ERROR)
32 }

```

```

1 /*-----*/
2 /* BUFFERED I/O - FOPEN */
3 /*-----*/
4
5 // opening file
6 #include <stdio.h>
7
8 int main()
9 {
10     // declare a new file handle
11     FILE* outFile = fopen("hello.txt", "w");
12     if (outFile == NULL)
13     {
14         fprintf(stderr, "ERROR: Unable to open
15             'hello.txt'\\n");
16         exit((8));
17     }
18     if (fprintf(outFile, "Hello World!\\n") <= 0)
19     {
20         fprintf(stderr, "ERROR: Unable to write to
21             'hello.txt'\\n");
22         exit(8);
23     }
24     return(0);
25 }
26
27 // general form fopen:
28 result = fopen(filename, mode);
29
30 // mode can be of the following:
31 r: read only
32 w: write only
33 r+: read and write
34 a: append (write but start at the end of file)
35 b: used in combination with the other modes for binary files
36
37 // syntax on mac & linux:
38 FILE* fopen("/root/file.txt", "w");
39
40 // syntax on win (backslash is the separator but \\r is return
41 // char, and \\f is the form char):
42 FILE* fopen("\\root\\file.txt", "w");

```

```

1 /*-----*/
2 /* BUFFERED I/O - FREAD & FWRITE & FFLUSH & FCLOSE */
3 /*-----*/
4
5 // reading binary file
6 // buffer is a pointer to the data buffer in which data placed
7 // elementSize is always 1; returns 0 for the end of the file
8 // returns negative if there is an error
9 // size of the buffer (number of bytes)
10 // inFile is the file to read
11 result = fread(buffer, elementSize, size, inFile);
12 result = fwrite(buffer, elementSize, size, inFile);
13
14 // copy infile.bin to outfile.bin
15
16 #include <stdio.h>
17 #include <stdlib.h>
18 #include <stdbool.h>
19
20 int main()
21 {
22     // the input file
23     // rb mode; r: read; b: binary
24     FILE* inFile = fopen("infile.bin", "rb");
25     if (inFile == NULL)
26     {
27         fprintf(stderr, "ERROR: Could not open onfile.bin\\n");
28         exit(8);
29     }
30
31     // the output file
32     FILE* outFile = fopen("outfile.bin", "wb");
33     if (outFile == NULL)
34     {
35         fprintf(stderr, "ERROR: Could not create
36             outfile.bin\\n");
37         exit(8);
38     }
39
40     // data buffer
41     char buffer[512];
42

```

```

43 while (true)
44 {
45     // return value is ssize_t: standard type that is
46     // big enough to hold
47     // the size of the largest object
48     // (structure, array, union)
49     // it also holds -1 for error condition)
50     ssize_t readSize = fread(buffer, 1, sizeof(buffer)
51     inFile);
52     if (readSize < 0)
53     {
54         fprintf(stderr, "ERROR: Read error seen\n");
55         exit(8);
56     }
57     if (readSize == 0)
58     {
59         break;
60     }
61     // returns a size_t value
62     // it is an unsigned type holds the size of the
63     // largest object
64     // it cannot hold an error value
65     // need casting between signed and unsigned
66     // types (size_t)readSize
67     if (fwrite(buffer, 1, readSize, outFile) !=
68     (size_t)readSize)
69     {
70         fprintf(stderr, "ERROR: Write error seen\n");
71         exit(8);
72     }
73     }
74     fclose(inFile);
75     fclose(outFile);
76     return (0);
77 }
78 // write the buffered data out now; ensures that data can be
79 // seen
80 printf("Before divide ");
81 fflush(stdout);
82 // close the file
83 int result = fclose(file);

```

```

1 /*-----*/
2 /* COMMND LINE ARGMENTS */
3 /*-----*/
4
5 // print the command line arguments
6 #include <stdio.h>
7
8 int main(const int argc, const char* argv[])
9 {
10     for (int i = 0; i < argc; ++i)
11     {
12         printf("argv[%d] = %s\n", i, argv[i]);
13     }
14     return (0);
15 }
16
17 $ ./prog first second third
18
19 argc    4
20 argv[0] ./prog
21 argv[1] first
22 argv[2] second
23 argv[3] third

```

```

1 /*-----*/
2 /* RAW I/O */
3 /*-----*/
4
5 // copy one file to another using buffer size of 1024 bytes
6 #include <stdio.h>
7 #include <stdbool.h>
8 #include <stdlib.h>
9 #include <unistd.h>
10 #include <sys/types.h>
11 #include <sys/stat.h>
12 #include <fcntl.h>
13
14 // conditional compilation
15 // linux does not have a O_BINARY flag but macos/win do have
16 // checks wether the O_BINARY is not defined; linux it isn't
17 // if os has that #define won't be compiled
18 #ifndef O_BINARY
19 // define O_BINARY with 0 value if not defined (for linux)
20 #define O_BINARY 0
21 #endif // O_BINARY

```

```

22 int main(int argc, char* argv[])
23 {
24     if (argc != 3)
25     {
26         fprintf(stderr, "Usage is %s <infile> <outfile>\n",
27             argv[0]);
28         exit(8);
29     }
30
31     // the fd (file-descriptor) of the input file
32     // fd = open(filename, flags)
33     // flags indicate how the input file is to be opened
34     // O_RDONLY flag opens the input file read-only
35     // O_BINARY flag indicates that the input file is binary
36     // don't use text files - not compatible between oss
37     int inFd = open(argv[1], O_RDONLY|O_BINARY);
38     if (inFd < 0)
39     {
40         fprintf(stderr, "ERROR: Could not open %s for
41             input\n", argv[1]);
42         exit(8);
43     }
44
45     // the fd (file-descriptor) of the output file
46     // fd = open(filename, flags)
47     // flags indicate how the output file is to be opened
48     // O_WRONLY flag opens the output file write only
49     // O_CREAT flag creates the file if needed
50     // O_BINARY flag indicates that the output file is binary
51
52     // 0666 is an octal number each digit representing a
53     // protection user set and each bit a protection type
54
55     // 1st user read and write (6) <user>
56     // 2nd accounts are in the same group as the user get
57     // read /write access (6) <group>
58     // 3rd anyone else gets the same read/write
59     // permission (6) <other>
60     int outFd = open(argv[2], O_WRONLY|O_CREAT|O_BINARY,
61         0666);
62     if (outFd < 0)
63     {
64         fprintf(stderr, "ERROR: Could not open %s for
65             writing\n", argv[2]);
66         exit(8);
67     }
68
69     while (true)
70     {
71         // buffer to read and write
72         char buffer[1024];
73
74         // size of the last read
75         size_t readSize;
76
77         // once the file open do the copy
78         // bytes_read = read(fd, buffer, size);
79         // size is the maximum number of characters read
80         // if that's negative it indicates an error
81         readSize = read(inFd, buffer, sizeof(buffer));
82
83         // check for an error
84         if (readSize < 0)
85         {
86             fprintf(stderr, "ERROR: Read error for file
87                 %s\n", argv[1]);
88         }
89
90         // check wether reached the end of the line and
91         // done transferring data
92         if (readSize == 0)
93             break;
94
95         // write that data
96         // bytes_written = write(fd, buffer, size);
97
98         // check for error
99         if (write(outFd, buffer, readSize) != readSize)
100         {
101             fprintf(stderr, "ERROR: Write error for %s\n",
102                 argv[2]);
103             exit(8);
104         }
105     }
106
107     // close the file descriptors
108     close(inFd);
109     close(outFd);
110     return (0);
111 }
112

```

```

113
114 $ ./copy input-file output-file

```

```

1 /*-----*/
2 /* FLOATING-POINT */
3 /*-----*/
4
5 // used in scientific or 3d graphics but not in embedded
6 // programming
7 // 1.0 = 1.
8 // 1.0e33 = 1.0 x 10^33
9 // float (single prec), double (double prec), long double (
10 // more precise)
11 // floating point constant
12 // F suffix: makes double to a single-precision float
13 // L suffix: makes float a long double
14 // decimal point is required otherwise this is integer divide
15 float f1 = 1/3;
16 0.0
17 float f2 = 1.0/3.0;
18 0.3333
19 // sign (+), fraction (four digits), exponent (e+56)
20 +1.234e+56
21
22 // numerical analysis and IEEE-754 deals with floating-point
23 // numbers
24 // floating point operations takes 1000 times longer than
25 // integer
26 // counterparts using libraries with no native support
27 // better chips with native support still calculates 10 times
28 // longer
29 // alternative - fixed point number
30
31 12.34    1234
32 00.01    1
33 12.00    1200
34 ...

```

```

1 /*-----*/
2 /* MODULAR */
3 /*-----*/
4
5 /*-----bad_example-----*/
6 // main.c
7 #include <stdio.h>
8
9 // extern keywords tells that the function is another file
10 // it does not always match the actual declaration (don't use
11 // it)
12 extern void funct(void);
13 int main()
14 {
15     printf("In main ()\n");
16     funct();
17     return (0);
18 }
19
20 // func.c
21 #include <stdio.h>
22 void funct(void)
23 {
24     printf("In funct()\n");
25 }
26
27 // makefile
28 // main must be rebuilt if main.c or func.c changes
29 main: main.c func.c
30 // compile both files and use them to make the program
31 gcc -g -Wall -Wextra -o main main.c
32 func.c
33
34 /*-----good_example-----*/
35 // main.c
36 #include <stdio.h>
37 // quotation marks indicate that the file to be included is
38 // user generated
39 // compiler will search for it in the current directory
40 // instead of searching through the system files
41 // inclusion provide the definition of the function
42 #include "func.h"
43 int main()
44 {
45     printf("In main()\n");
46     funct();
47     return (0);
48 }
49

```

```

48 // func.c
49 #include <stdio.h>
50 // compiler check the definition of the function
51 #include "func.h"
52 void funct(void)
53 {
54     printf("In funct()\n");
55 }
56
57 // create a header file to hold the extern definition
58 // don't need to add extern function funct in several diff
59 // files
60 // #ifndef/#endif is double inclusion protection (if funct is
61 // in
62 // multiple header files).h
63 #ifndef __FUNC_H__
64 #define __FUNC_H__
65 extern void funct(void);
66 #endif // __FUNC_H__
67
68 // makefile
69 // compile program macro
70 CFLAGS = -g -Wall -Wextra
71 // OBJ macro contains list of objects used to make the
72 // program
73 OBJS = main.o func.o
74 main: $(OBJS)
75 gcc -g -Wall -Wextra -o main $(OBJS)
76 // create main.o from main.c and func.h
77 main.o: main.c func.h
78 func.o: func.c func.h
79
80 // rules:
81
82 // each module should have a header file with the same name
83 // as the module
84 // header file should contain the definitions of the public
85 // types,
86 // variables, and functions and nothing else
87 // every module should include its own header file so C can
88 // check
89 // to make sure the header file and implementation match
90 // modules should include code used for a common purpose
91 // modules should expose minimum information into the outside
92 // information modules expose via extern declarations is
93 // global
94 // (seen by the entire program)
95
96 // namespaces - no namespaces in C; no function symbol
97 // duplication is allowed; prefixes are used;
98 // HAL_StatusTypeDef; it means StatusTypeDef belongs to HAL
99 // library

```

3 config

```
1 /*-----*/
2 /* NVIM */
3 /*-----*/
4
5 // show line numbers automatically
6 $ ~/.config/nvim
7 $ nvim init.vim
8 source ~/.vimrc
9 $ ~/
10 $ nvim .vimrc
11 set number
12
13 /*-----*/
14 /* TMUX */
15 /*-----*/
16
17 // ~/.tmux.conf
18 unbind C-Space
19 set -g prefix C-Space
20 bind C-Space send-prefix
21 set -g mouse on
22 set-option -g history-limit 5000
23
24 /*-----*/
25 /* SSH */
26 /*-----*/
27
28 // ~/.ssh/config
29 $ cat ~/.ssh/config
30 Host name
31   User foo
32   Hostname 127.0.0.1
33   Port 8765
34 $ ssh name
35
36 /*-----*/
37 /* MAKE */
38 /*-----*/
39
40 // Makefile
41 CFLAGS=-Wall -g
42 clean:
43   rm -f ex1
44
45
46 1 /*-----*/
47 2 /* GIT */
48 3 /*-----*/
49 4
50 5 $ git config --global user.name "marcellgyorei"
51 6 $ git config --global user.email "marcell.gyorei@gmail.com"
52 7 $ git config --global color.ui true
53 8 $ git config --global core.editor nvim
54 9
55 10 // config values
56 11 nano          nano
57 12 vim           vim
58 13 neovim        nvim
59 14 emacs         emacs
60 15 sublime text  subl -n -w
61 16 atom          atom --wait
62 17 vscode       code --wait
63 18
64 19 // create keygen in ~/.ssh folder
65 20 // id_rsa & id_rsa.pub files will be created
66 21 $ ssh-keygen -t rsa -C "marcell.gyorei@gmail.com"
67 22
68 23 // github.com/Account Settings/SSH Keys
69 24 // Add SSH Key ("My laptop")
70 25 // copy ssh public key into the given box
71 26
72 27 // test connection
73 28 $ ssh -T git@github.com
74 29
75 30 // check if SSH key fingerprint matching with public ones
76 31 Hi username! You've successfully authenticated ..
77
78
79 1 /*-----*/
80 2 /* GIT-CRESCENT */
81 3 /*-----*/
82 4
83 5 // keygen folder on crescent
84 6 /scratch/s392494/.ssh/id_rsa.pub
85 7
86 8 // go back into root
87 9 cd ~
```