

## bash + make + git + vim

```

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
60 $ pwd # get location of current dir
61 $ find /root/sid/ -name "*matrix*" # search for file
62 $ rm -rf spmv_openmp # force remove
63 $ cp -R t1/. t2/ # copy content

```

```

1 #-----#
2 # MAKE #
3 #-----#
4
5 # compiling with linking in non-default name '-o'
6 # read.o is dependency
7 # if timestamp 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
75 # delete all local changes
76 git reset --hard

```

```

1  /-----*/
2  /* VIM_MODE */
3  /-----*/
4
5  save as exl          :w! exl
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

```

## vscode + cmake + llbd + catch

```

1  /-----*/
2  /* 0.1-KEYBINDINGS */
3  /-----*/
4
5  $ S-C-e              # explorer
6  $ Cl-^              # terminal
7  $ cd build && cmake .. && make # build make files and ex
8  /* or */
9  $ F7                # build make files [cmake]
10 $ C-S-b             # build task [cmake]
11 /* */
12 $ S-C-d             # run & debug
13 $ S-C-p             # command palette
14
15 /-----*/
16 /* 0.2-CONFIG */
17 /-----*/
18
19 # CMakeLists.txt      # cmake config [make]
20 # .vscode/c_cpp_properties.json # intellisense config
21 # .vscode/tasks.json  # task built config [cmake]
22 # .vscode/launch.json # run & debug [llbd+task]
23 # build/compile_command.json # compiler commands [clang++]
24
25 /-----*/
26 /* 1-SETUP WORKSPACE CREATE SOURCE FILES */
27 /-----*/
28
29
30 $ C-o              # add new folder to workspace
31 $ C-k f           # close workspace
32 $ touch main.cpp  # add new file [bash] Cl-O-C-n
33 $ S-C-f           # search
34 $ C-s             # save

```

```

1  /-----*/
2  /* 2.1-GENERATE CMakeList.txt */
3  /-----*/
4
5  $ S-C-p
6  ^ CMAKE:Quick Start
7  ^ Clang 10.0.0
8  ^ projectname
9  ^ Executable
10
11 $ F7                # cmake build from
12                   # compile_commands.json
13 // or
14 $ cd build $$ cmake .. # build make files
15 $ make              # build executable

```

```

1  /-----*/
2  /* 2.2-ADD SOURCE FILES TO CMakeList.txt */
3  /-----*/
4
5  /* to generate Make files using CMake */
6
7  cmake_minimum_required(VERSION 3.0.0)
8  project(pcmake VERSION 0.1.0)
9  include(CTest)
10 enable_testing()
11 add_executable(HelloWorld main.cpp Pet.cpp)
12 set(CPACK_PROJECT_NAME ${PROJECT_NAME})

```

```

13 set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
14 include(CPack)

```

```

1 /*-----*/
2 /* 3.1-SETUP Intellisense */
3 /*-----*/
4
5 ~ C/C++:Edit Configuration (JSON)
6 ~ c_cpp_properties.json

```

```

1 /*-----*/
2 /* 3.2-CHECK c_cpp_properties.json */
3 /*-----*/
4
5 {
6   "configurations": [
7   {
8     "name": "Mac",
9     "includePath": [
10       "${workspaceFolder}/**",
11       // "${workspaceFolder}/include"
12     ],
13     "defines": [],
14     "macFrameworkPath": [
15       "/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/
16       System/Library/Frameworks"
17     ],
18     "compilerPath": "/usr/bin/clang",
19     "cStandard": "c17",
20     "cppStandard": "c++17",
21     "intelliSenseMode": "macos-clang-arm64",
22     "configurationProvider": "ms-vscode.cmake-tools"
23   }
24 ],
25 "version": 4
26 }

```

```

1 /*-----*/
2 /* 4.1-GENERATE CMake-Make TASK */
3 /*-----*/
4
5 $ touch ../.vscode/tasks.json

```

```

1 /*-----*/
2 /* 4.2-INSERT CMake TASK COMMANDS INTO task.json */
3 /*-----*/
4
5 /* assign CMake build process to a VSC task */
6
7 {
8   // See https://go.microsoft.com/fwlink/?LinkId=733558
9   // for the documentation about the tasks.json format
10   "version": "2.0.0",
11   "isShellCommand": true,
12   "options": {
13     "cwd": "${workspaceRoot}/build" // workspaceRoot is the
14                                     // directory
15                                     // of the workspace
16   },
17   "tasks": [
18   {
19     "label": "cmake", // allows vsc to reference the
20                       // task name when running the
21                       // task
22     "command": "cmake -G 'Unix Makefiles' -DCMAKE_BUILD_TYPE=Debug
23               ..", "type": "shell",
24     "presentation": {
25       "echo": true,
26       "reveal": "always",
27       "panel": "shared"
28     }
29   },
30   {
31     "label": "make",
32     "command": "make -j 8", // it means running the
33                           // compiler in parallel with
34                           // max 8 source files
35     "presentation": {
36       "echo": true,
37       "reveal": "always",
38       "panel": "shared"
39     },
40     "isBuildCommand": true // if true to be executed with
41                           // Tasks: Run Build Task
42   }
43 ]
44 }

```

```

1 /*-----*/
2 /* RUN TASK */
3 /*-----*/
4
5 ~ Run Task
6 ~ cmake
7 ~ enter
8
9 $ C-S-b
10
11 // check
12 $ .build/pcmake

```

```

1 /*-----*/
2 /* GENERATE launch.json */
3 /*-----*/
4
5 $ touch launch.json

```

```

1 /*-----*/
2 /* MODIFY launch.json */
3 /*-----*/
4
5 /* assign VSC task and lldb to run & debug */
6
7 {
8   // Use IntelliSense to learn about possible attributes.
9   // Hover to view descriptions of existing attributes.
10   // For more information, visit: https://go.microsoft.com/
11   // fwlink/?linkid=830387
12   "version": "0.2.0",
13   "configurations": [
14   {
15     "name": "(lldb) Launch",
16     "type": "cppdbg",
17     "request": "launch",
18     "program": "${workspaceFolder}/build/pcmake",
19     "args": [],
20     "stopAtEntry": false,
21     "cwd": "${workspaceFolder}",
22     "environment": [],
23     "externalConsole": true,
24     "MIMode": "lldb"
25   }
26 ]
27 }

```

```

1 /*-----*/
2 /* CHECK lldb */
3 /*-----*/
4
5 $ which lldb
6 $ F5

```

```

1 /*-----*/
2 /* SETUP GOOGLETEST */
3 /*-----*/
4
5 # C++_1.38__Introduction to Google Test and CMake
6 # https://www.youtube.com/watch?v=0XwsD37qHPY
7
8 $ C-k f
9 $ mkdir gexample
10 $ C-o
11 $ git clone https://github.com/google/googletest.git
12 $ touch CMakeList.txt

```

```

1 cmake_minimum_required(VERSION 3.8)
2
3 set(gexample #[[project name]])
4
5 project(${This} C CXX)
6
7 set(CMAKE_C_STANDARD 17)
8 set(CMAKE_CXX_STANDARD 17)
9 set(CMAKE_POSITION_INDEPENDENT_CODE ON) #[[libraries & ex
10   mixed]]
11
12 enable_testing() #[[have unit test]]
13
14 add_subdirectory(googletest) #[[add dependency]]
15
16 #[[variables defining what will be built]]
17 set(Headers
18   Example.hpp
19 )
20 set(Sources
21   Example.cpp
22 )

```

```

22 # [[link other things to for programs - code is the library
23     that is tested and statically linked, gtest is the
24     program]]
25 add_library(${This} STATIC ${Sources} ${Headers})
26 # [[put the test into a subdirectory; it looks for another
27     CMakeLists.txt and runs that in a child node in the build
28     system]]
29 add_subdirectory(test)

```

```

1 $ touch Example.cpp
2 $ touch Example.hpp
3 $ mkdir test
4 $ touch CMakeLists.txt

```

```

1 cmake_minimum_required(VERSION 3.8)
2
3 set(gexampletests)
4
5 # [[no need for headers for tests]]
6 set(Sources
7     ExampleTests.cpp
8 )
9 # [[instead of making linked to the program we are making the
10    program itself]]
11 add_executable(${This} ${Sources})
12 # [[linking in the main program into the test provided by
13    googletest]]
14 # [[need the thing testing]]
15 # [[clue for the compiler where to find header files]]
16 target_link_libraries(${This} PUBLIC
17     gtest_main
18     Example
19 )
20 # [[this is a unit test]]
21 add_test(
22     NAME ${This}
23     COMMAND ${This}
24 )

```

```

1 $ ExampleTests.cpp
2 ^ reload
3 # builds the build system after choosing a compiler
4 ^ Clang 5.01
5 # builds the executables
6 $ F7
7
8 # fix built-vscode directory name
9 ^ Preferences: OpenSettings (JSON)
10 ^ USER SETTINGS
11 ^ "cmake.buildDirectory": "${workspaceroot}/build", #line22
12
13 # fix wether linked statically or dynamically (should be
14    linked statically)
15 ^ build/CMakeCache.txt
16 ^ search: lib #line294
17 ^ gtest_force_shared_crt:BOOL=ON
18 ^ Clean rebuild #under CMake tab #find shortcut later to CMake
19    Clean rebuild
20
21 # run the test for vscode
22 ^ CMake: Run tests

```

```

1 /*-----*/
2 /* FILL ExampleTests.cpp */
3 /*-----*/
4
5 #include <gtest/gtest.h>
6
7 bool f() {
8     return false;
9 }
10
11 TEST(gexampletests, DemonstrateGtestMacros)
12     // sanity checking
13     EXPECT_TRUE(false);
14     // if we don't pass the line don't continue
15     ASSERT_TRUE(false);
16     EXPECT_EQ(true, true);
17
18     // expected value first and actual value second
19     const bool result = f();
20     EXPECT_EQ(true, result); // that won't pass as return =
21     false
22 }
23 // 15:55

```

# 1 c++

```

1  /*-----*/
2  /* LEARNING SOURCES */
3  /*-----*/
4
5  /* REAL-TIME */
6
7  1st C++_2.16__E_Christopher Kormanyos - Real-Time C++ 2021
8
9  /* CLARITY */
10
11  2nd C++_2.18__E_Rainer - C++ Core Guidelines Explained 2022
12  /**/
13  2nd C++_1.23__Jason Turner - C++ Best Practices 2020
14  ref C++_1.34__S_ISO_IEC 14882_2020 Sixth edition 2020
15  ref C++_2.17__S_Bjarne - CppCoreGuidelines
16
17  /* CORE BASICS 2020 & STL */
18
19  2nd C++_1.36__E_Pitt - Guide to Scientific Computing 2018
20  /**/
21  ref C++_1.37__Bjarne - A Tour of C++ 2018
22  ref C++_1.38__Bjarne - PPP C++ 2021
23  /**/
24  ref C++_1.31__E_Paul Deitel - C++ for Programmers 2022
25  ref C++_1.28__E_Hacking C++ - C++ Cheat Sheets & Infograph STL
26
27  /* LOW BUILT-TIME */
28  /* DATA STRUCTURES, ALGORITHMS */
29
30  C++_2.7__S_Joe Gibson - C++ Data Str & Algo Cheat Sheet
31
32  /* TEST */
33
34  C++_2.15__E_Leetcode - C++
35
36  /* TDD */
37
38  C++_1.38__Introduction to Google Test and CMake
39
40  /* COMPETITIVE PROGRAMMING */
41
42  /* REQS, DESIGN, SPECS */
43  /* EMBEDDED CPS */
44  Edward Ashford Lee - Introduction to Embedded Systems 2017
45  UseCaseDiagram
46  StructureModelling
47  // UML paperback
48  // omnigraffle/window/stencil/search uml
49  // sketch
50
51  /* WHITE-BOARDING */
52  /* DESIGN TOOL */
53  /* TRADING TOOL */
54
55
56  /* ONLINE POOL */
57  STRATEGY
58  // https://medium.com/@alexander.s.augenstein/how-i-passed
59  -the-c-code-interview-in-3-weeks-a3e350214a01
60
61  PROBLEM SOLVING PATTERNS
62  // https://www.designgurus.io/blog/dont-just-leetcode

```

```

1  /* DOCTEST */
2
3  #define DOCTEST_CONFIG_IMPLEMENT_WITH_MAIN
4  #include <doctest.h>
5
6  int fact(int n) { return n <= 1 ? n : fact(n-1) * n; }
7
8  TEST_CASE("testing the factorial function") {
9      CHECK(fact(0) == 1); // will fail
10     CHECK(fact(1) == 1);
11     CHECK(fact(2) == 2);
12     CHECK(fact(10) == 3628800);
13 }
14 // CppCon 2017

```

```

1  /* PPP C++ */
2  /* #17 VECTOR, POINTERS */
3
4

```

```

1  /* TURNER - C++ BEST PRACTICES 2020 */
2  // #18
3
4  float divide(float numerator, float denominator)

```

```

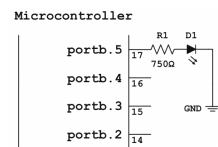
5  {
6      return numerator / denominator;
7  }
8
9  int divide(int numerator, int denominator)
10 {
11     return numerator / denominator;
12 }
13
14 template<typename Arithmetic>
15 Arithmetic divide(Arithmetic numerator, Arithmetic denominator)
16 {
17     return numerator / denominator;
18 }
19
20 // lambda alternative

```

```

1  /* #35 STRONG TYPES (NO BOOLEAN ARGUMENTS) */
2
3  struct Widget
4  {
5      enum struct Visible { True, False };
6      enum struct Resizable { True, False };
7
8      Widget(Visible visible, Resizable resizable);
9  }

```



```

1  /* KORMANYOS - REAL-TIME C++ 2021 */
2  /* LED PROGRAM [PAGE 4] */
3
4  #include <csdint>
5  #include "mcal_reg.h"
6
7  class led
8  {
9  public:
10     typedef std::uint8_t port_type;
11     typedef std::uint8_t bval_type;
12
13     led(const port_type p,
14         const bval_type b) : port(p),
15                             bval(b)

```

```

1  /* COMPOUND ASSIGNMENT */
2
3

```

```

1  /* OBJECTS & INSTANCES [PAGE 12] */
2
3  const led led_b5
4  {
5      mcal::reg::portb,
6      mcal::reg::bval5
7  };
8
9  // led_b5 is an instance of the led class
10 // parameters in the constructor of led_b5 use uniform
11 // initialization syntax
12 // led_b5 is a constant object that wont be modified for the
13 // entire lifetime of the program
14 // compiler initialize automatically the led_b5 static
15 // instance before would be used in main() - this called
16 // startup code

```

```

1  // led.h
2  class led
3  {
4  public:
5      led(const port_type p,
6          const bval_type b);
7
8      void toggle() const;
9  };

```

```

1  // led.cpp

```

$$\vec{u} \cdot \vec{v} = uv$$

```

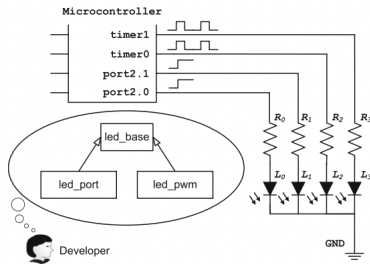
1  /* NUMERIC [PAGE 54] */
2  /* INNER PRODUCT */
3  #include <array>
4  #include <numeric>
5
6  const std::array<int, 3U> u
7  {
8      { 1, 2, 3 }
9  };
10
11 const std::array<int, 3U> v
12 {
13     { 4, 5, 6 }
14 };
15
16 const int uv = std::inner_product(u.begin(),
17                                   u.end(),
18                                   v.begin(),
19                                   0);
20 $ 32

```

```

1  /* RANDOM [PAGE 64] */

```



```

1  /* INHERITANCE [PAGE 76] */

```

```

1  /* CUSTOM MEMORY ALLOCATION [ON HEAP] [PAGE 238] */
2
3

```

```

1  /* RING ALLOCATOR [PAGE 244] */

```

## strategy

- Basic syntax
- Ranged for loops
- Templates
- Operator overloading
- Preprocessor statements and macros
- Namespaces
- Exceptions
- Const and mutable (and friend functions)
- Static and dynamic casting
- Pointers / void pointers / smart pointers
- Enums
- String streams
- Virtual functions and virtual inheritance and vttables
- Runtime type information (RTTI)
- Helpful 3rd party libraries (Qt, OpenGL, TensorFlow, OpenCV, Eigen, Drake)
- The Standard Library (STL) (containers, the algorithms over them, and the iterators that connect them)
- Visual studio debug tools (breakpoints, memory view)
- Package management
- g++
- gtest
- cmake

## best practice

- Memory allocation as initialization
- Dont use casts
- Avoid naked new and naked delete
- Avoid macros except for include guards
- Avoid unions
- Hide arrays from interfaces, keep them in low-level if needed
- Prefer immutable to mutable data
- Encapsulate messy constructs rather than spreading them thru the code
- Interfaces are the single most important aspect of code organization, make them explicit, encapsulate rule violations
- State preconditions, prefer Expects() for expressing preconditions and Ensures() for postconditions
- Prefer <vector> by default, prefer <array> to C-style arrays if needed
- C++98 added templates, containers, <algorithm>, <string>, <iostream>
- C++11 added auto, function pointers / lambdas, multithreading, regex, smart pointers, hash tables
- C++14 added read-write locks, generic lambdas //C++17 added parallel algorithms
- C++20 added the spaceship operator and templated lambdas

## 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, sqrtout;
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     sqrtout = gen_sqrt(val);
25
26     // print out the result
27     printf("The generalised square root of %lf is %lf\n", val,
28           sqrtout);
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/1)      (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                     default: printf("Incorrect choice\n");
38
39         }
40         return 0;
41     }
42
43 // action routines
44 void action1()
45 {
46     printf("This is the action1 routine\n");
47 }
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        if (disaster)
13            goto error;
14    ...
15 }
16
17 error:
18     /* error handling */
19     return;
20

```

```

1 /*-----*/
2 /* FUNCTION PROTOTYPES */
3 /*-----*/
4
5 // function definition
6 char func(int lower, int *upper, char (*func)(), double y)
7 {}
8
9 // prototype declaration v1
10 char func(int lower, int *upper, char (*func)(), double y);
11
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 predefined 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 char 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 // ampersands used because it needs to modify the arguments
26 // therefore arguments must be passed by address
27 // sscanf returns the number of items it converted
28 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\n\n");
15         exit(8);
16     }
17     if (fprintf(outFile, "Hello World!\n") <= 0)
18     {
19         fprintf(stderr, "ERROR: Unable to write to\n\n");
20         exit(8);
21     }
22     return(0);
23 }
24
25 // general form fopen:
26 result = fopen(filename, mode);
27
28 // mode can be of the following:
29 r: read only
30 w: write only
31 r+: read and write
32 a: append (write but start at the end of file)
33 b: used in combination with the other modes for binary files
34
35 // syntax on mac & linux:
36 FILE* fopen("/root/file.txt", "w");
37
38 // syntax on win (backslash is the separator but \r is return
39 char, and \f is the form char):
40 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\n\n");
36         exit(8);
37     }
38
39     // data buffer
40     char buffer[512];
41
42     while (true)
43     {
44         // return value is ssize_t: standard type that is
45         // big enough to hold
46         // the size of the largest object
47         // (structure, array, union)
48         // it also holds -1 for error condition)
49         ssize_t readSize = fread(buffer, 1, sizeof(buffer)
50             inFile);
51         if (readSize < 0)
52         {
53             fprintf(stderr, "ERROR: Read error seen\n");
54             exit(8);
55         }
56         if (readSize == 0)
57         {
58             break;
59         }
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
79 // write the buffered data out now; ensures that data can be
80 // seen
81 printf("Before divide ");
82 fflush(stdout);
83 // close the file
84 int result = fclose(file);

```

```

1 /*-----*/
2 /* COMMAND LINE ARGUMENTS */
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
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

```

```

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 whether 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
23 int main(int argc, char* argv[])
24 {
25     if (argc != 3)
26     {
27         fprintf(stderr, "Usage is %s <infile> <outfile>\n",
28             argv[0]);
29         exit(8);
30     }
31
32     // the fd (file-descriptor) of the input file
33     // fd = open(filename, flags)
34     // flags indicate how the input file is to be opened
35     // O_RDONLY flag opens the input file read-only
36     // O_BINARY flag indicates that the input file is binary
37     // don't use text files - not compatible between oss
38     int inFd = open(argv[1], O_RDONLY|O_BINARY);
39     if (inFd < 0)
40     {
41         fprintf(stderr, "ERROR: Could not open %s for
42             input\n", argv[1]);
43         exit(8);
44     }
45
46     // the fd (file-descriptor) of the output file
47     // fd = open(filename, flags)
48     // flags indicate how the output file is to be opened
49     // O_WRONLY flag opens the output file write only
50     // O_CREAT flag creates the file if needed
51     // O_BINARY flag indicates that the output file is binary
52
53     // 0666 is an octal number each digit representing a
54     // protection user set and each bit a protection type
55
56     // 1st user read and write (6) <user>
57     // 2nd accounts are in the same group as the user get
58     // read /write access (6) <group>
59     // 3rd anyone else gets the same read/write
60     // permission (6) <other>
61     int outFd = open(argv[2], O_WRONLY|O_CREAT|O_BINARY,
62         0666);
63     if (outFd < 0)
64     {
65         fprintf(stderr, "ERROR: Could not open %s for
66             writing\n", argv[2]);
67         exit(8);
68     }
69
70     while (true)
71     {
72         // buffer to read and write
73         char buffer[1024];
74
75         // size of the last read
76         size_t readSize;

```

```

77
78         // once the file open do the copy
79         // bytes_read = read(fd, buffer, size);
80         // size is the maximum number of characters read
81         // if that's negative it indicates an error
82         readSize = read(inFd, buffer, sizeof(buffer));
83
84         // check for an error
85         if (readSize < 0)
86         {
87             fprintf(stderr, "ERROR: Read error for file
88                 %s\n", argv[1]);
89         }
90
91         // check whether reached the end of the line and
92         // done transferring data
93         if (readSize == 0)
94             break;
95
96         // write that data
97         // bytes_written = write(fd, buffer, size);
98
99         // check for error
100        if (write(outFd, buffer, readSize) != readSize)
101        {
102            fprintf(stderr, "ERROR: Write error for %s\n",
103                argv[2]);
104            exit(8);
105        }
106    }
107
108    // close the file descriptors
109    close(inFd);
110    close(outFd);
111    return (0);
112 }
113
114 $ ./copy input-file output-file
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459

```

```

13 {
14     printf("In main ()\n");
15     funct();
16     return (0);
17 }
18
19 // func.c
20 #include <stdio.h>
21 void funct(void)
22 {
23     printf("In funct()\n");
24 }
25
26 // makefile
27 // main must be rebuilt if main.c or func.c changes
28 main: main.c func.c
29 // compile both files and use them to make the program
30 gcc -g -Wall -Wextra -o main main.c
31 func.c
32
33 /*-----good_example-----*/
34 // main.c
35 #include <stdio.h>
36 // quotation marks indicate that the file to be included is
37 // user generated
38 // compiler will search for it in the current directory
39 // instead of searching through the system files
40 // inclusion provide the definition of the function
41 #include "func.h"
42 int main()
43 {
44     printf("In main()\n");
45     funct();
46     return (0);
47 }
48
49 // func.c
50 #include <stdio.h>
51 // compiler check the definition of the function
52 #include "func.h"
53 void funct(void)
54 {
55     printf("In funct()\n");
56 }
57
58 // create a header file to hold the extern definition
59 // don't need to add extern function funct in several diff
60 // files
61 // #ifnded/#endif is double inclusion protection (if funct is
62 // in
63 // multiple header files).h
64 #ifndef __FUNC_H__
65 #define __FUNC_H__
66 extern void funct(void);
67 #endif // __FUNC_H__
68
69 // makefile
70 // compile program macro
71 CFLAGS = -g -Wall -Wextra
72 // OBJ macro contains list of objects used to make the
73 // program
74 OBJJS = main.o func.o
75 main: $(OBJJS)
76 gcc -g -Wall -Wextra -o main $(OBJJS)
77 // create main.o from main.c and func.h
78 main.o: main.c fun.h
79 func.o: func.c func.h
80
81 // rules:
82
83 // each module should have a header file with the same name
84 // as the module
85 // header file should contain the definitions of the public
86 // types,
87 // variables, and functions and nothing else
88 // every module should include its own header file so C can
89 // check
90 // to make sure the header file and implementation match
91 // modules should include code used for a common purpose
92 // modules should expose minimum information into the outside
93 // information modules expose via extern declarations is
94 // global
95 // (seen by the entire program)
96
97 // namespaces - no namespaces in C; no function symbol
98 // duplication is allowed; prefixes are used;
99 // HAL_StatusTypeDef; it means StatusTypeDef belongs to HAL
100 // 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 /*-----*/
47 /* GIT */
48 /*-----*/
49
50 $ git config --global user.name "marcellgyorei"
51 $ git config --global user.email "marcell.gyorei@gmail.com"
52 $ git config --global color.ui true
53 $ git config --global core.editor nvim
54
55 // config values
56 nano          nano
57 vim           vim
58 neovim        nvim
59 emacs         emacs
60 sublime text  subl -n -w
61 atom          atom --wait
62 vscode        code --wait
63
64 // create keygen in ~/.ssh folder
65 // id_rsa & id_rsa.pub files will be created
66 $ ssh-keygen -t rsa -C "marcell.gyorei@gmail.com"
67
68 // github.com/Account Settings/SSH Keys
69 // Add SSH Key ("My laptop")
70 // copy ssh public key into the given box
71
72 // test connection
73 $ ssh -T git@github.com
74
75 // check if SSH key fingerprint matching with public ones
76 Hi username! You've successfully authenticated ..
77
78
79 /*-----*/
80 /* GIT-CRESCENT */
81 /*-----*/
82
83 // keygen folder on crescent
84 /scratch/s392494/.ssh/id_rsa.pub
85
86 // go back into root
87 cd -
88

```

## notes

```
1  /*-----*/
2  /* APPENDIX - LEARNING SOURCES */
3  /*-----*/
4
5  /* SUPPLEMENTARY */
6
7  // https://cplusplus.com/doc/tutorial/functions/
8  // C++_4.12__Eijkhout - Intro to Scientific Pr in C++ 2022
9  // C++_1.24__Introduction to C++ MIT_6.096 2011
10 // C++_1.35__E_Marc Gregoire - Professional C++ 2021 [Int
    Ch]
11 // xB_1.13__E_Gayle - Cracking the Coding Interview 2015
12 // C++_2.8__Adnan Aziz - Elements of Programming Int 2015
13 // xB_1.12__E_Georgevich - The Complete Interview Answer
14 // Peter Sherar - N-CST-SSPP_ C++ Programming
15 // keywords: linked compilation
16 // C++_2.5__INT-S_Pat Morin - Open Data Structures.pdf
17 // Siemens - Parasolid kernel
18 // Siemens - D-Cubed constraint solver
```

## 4 design

```
1
2  Cyber-Physical System Approach
3
4  -Modelling [Dynamics = Evolution in Time]
5
6  -Actor Models [Continuous-time systems with
    Feedback Control; all modelled as Functions]:
7
8      --- [Stage 0]
9      -Project [Logical]
10     --- [Stage 1]
11     -Context [Logical]
12     --- [Stage 2]
13     -Operation [Logical]
14     --- [Stage 2-3-4]
15     -System [Logical & Physical]
16     --- [Stage 5]
17     -Assembly [Physical]
18     -Product [Physical]
19
20     -Abstract to High Fidelity Representations of
    dynamics and static properties of Actor Models
21     -Hierarchical State Machines
22     -Supervisory Control [Abstract Process on 5 Reps]
23     -Dataflow Models
24     -Time Models
25
26     -Design
27     -Memory Architectures
28     -Multitasking
29     -Scheduling
30
31     -Analysis
32     -Requirements of Hierarchical State Machines
```