Полезные инструкции по установке программ для школы 21

Полезные инструкции

на экзамене можно было и VScode и Xcode открыть
open -a "Visual Studio Code"
и так же с Xcode

— —
Переключиться с/на bash, zsh
exec bash
exec zsh

Параметры вызываемых программ
norminette -R CheckForbiddenSourceHeader
norminette -R CheckDefine
gcc -Wall -Wextra -Werror

:term - открыть терминал в vim
Ctrl + w - переключаться между окнами

-----
https://kuzevanov.ru/macos/настройка-zsh-в-mac-os.html
https://robotrainer.github.io/school21/#gcc
https://qastack.ru/programming/18428374/commands-not-found-on-zsh
https://github.com/42School/norminette

Отключить сглаживание шрифтов
defaults -currentHost write -g AppleFontSmoothing -int 0
(0- отключение сглаживания
1- легкое сглаживание - я выбрал это
2- средний уровень сглаживания
3- сильный)
Перезагрузить комп

Установить программы:
---
1. Xcode
1.1. Xcode (AppStore) - 11 GB!
1.2. Запустить Xcode и принять лицензию
-----
2. ITerm2 (https://iterm2.com)
-----
3. Python (https://www.python.org/downloads/)
-----

4. Visual Studio Code

4.1 Visual Studio Code (https://code.visualstudio.com/docs/?dv=osx)
4.2 Настроить VSC:
Code-Preferences-Settings
В поиске пишем Tab
Убираем тики: Detect Indentation, Insert Spaces
---

5. Norminette

5.1 Установить Норминет:
python3 -m pip install norminette --user

5.2 В процессе установки появится предупреждение:
Installing collected packages: norminette
  WARNING: The script norminette is installed in '/Users/dmitry/Library/Python/3.8/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed norminette-3.3.1

5.3 Согласно предупреждению из п.5.2 необходимо прописать путь к Норминету:
(https://www.architectryan.com/2012/10/02/add-to-the-path-on-mac-os-x-mountain-lion/)

Open up Terminal.
Run the following command:
sudo nano /etc/paths
Enter your password, when prompted.
Go to the bottom of the file, and enter the path you wish to add. (/Users/dmitry/Library/Python/3.8/bin)
Hit control-x to quit.
Enter “Y” to save the modified buffer.
That’s it!

!!!To test it, in !!!new terminal window, type:
echo $PATH
You should see something similar to this (including the path you’ve added!):
dmitry@d-mac ~ % echo $PATH
/Library/Frameworks/Python.framework/Versions/3.9/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/Users/dmitry/Library/Python/3.8/bin:/Library/Apple/usr/bin

5.4 В папке проекта в терминале вызвать: norminette
Должно работать.
---

6. Install Keynote, Numbers, Pages (AppStore)

---

7. Header: (https://robotrainer.github.io/school21/#gcc)

Во всех файлах должен быть заголовок (header), который установлен правилами Школы
Для создания заголовка 42header необходимо использовать плагин, который скачивается и устанавливается с помощью пакетного менеджера Vundle.

Плагин нужен только тем, кто работает на своём пк/ноутбуке. На компьютерах школы всё и так прекрасно работает.

Создадим необходимую директорию и скачаем туда менеджер Vundle:

```
$ mkdir -p ~/.vim/bundle
$ git clone https://github.com/VundleVim/Vundle.vim.git ~/.vim/bundle/Vundle.vim
```

Далее настраиваем Vim и пактный менеджер Vundle, а также запишем в очередь на установку плагин 42header.

Для этого создадим в каталоге ~/.vim файл vimrs:
$ cd .vim/
$ vim vimrc

Был создан и сразу открыт в редакторе vim файл vimrc, в который необходимо записать следующие настройки
(последняя строка делает печать заголовка при нажатии на кнопку F5):

```
set number
set cursorline
set cursorcolumn
set autoindent
set cindent
highlight ExtraWhitespace ctermbg=red guibg=red
match ExtraWhitespace /\s\+$\|\s+\s{1}/
highlight MoreThan80 ctermbg=blue guibg=blue
:2match MoreThan80 /\%81v.\+/
set tabstop=4
set shiftwidth=4

set nocompatible
filetype off
set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()
Plugin 'VundleVim/Vundle.vim'
Plugin 'pandark/42header.vim'

" Add plugins here

call vundle#end()
filetype plugin indent on
nmap <f5> :FortyTwoHeader<CR
```

Не забываем сохранить изменения и выйти из vim командой :wq.

Теперь необходимо установить и обновить прописанные в vimrc плагины. Откройте vim и выполните в нём команду:
$ vim
:PluginInstall
:qa

Пропишем в файле настроек командной оболочки имя пользователя $USER и почту $MAIL для нашего заголовка 42header.
Открываем .bashrc (или .zshrc, если у вас zsh):
$ vim .bashrc

и прописываем в конце строки:
export USER=тут_будет_ваш_login
export MAIL=тут_будет_ваш_email

После сохранения .bashrc, его необходимо перезапустить:
$ source .bashrc

Проверяем.
Запускаем vim и жмем кнопку F5 -> Enter

Другая проверка: Запускаем vim и прописываем команду:
:FortyTwoHeader -> Enter
---

8. Настроить git

В папке проектов
```
git init
```

Открыть для редактирования
```
vim .gitignore
```

Прописать в нем:
```
.gitignore
.DS_Store
a.out
:wq
```

Склонировать в папку проектОВ свой репозитарий, например
```
git clone https://github.com/Dmitry-GDG/school21.git
```

Перейти в папку склонированного проекта
```
git status
git add *  //возможно, придется добавить отдельно каждую папку
git status
git commit -m "my first commit"
git status
git push
```
---

9. Install gdb - НЕ ПОЛУЧИЛОСЬ (Не понял, как работает)

https://dev.to/jasonelwood/setup-gdb-on-macos-in-2020-489k

9.0 Проверяем, установлен ли:
```
gdb —version
```
Если получили ошибку, тогда проверяем, установлен ли Homebrew
```
brew --version
```
Если получили ошибку, тогда необходимо установить Homebrew:
```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)
```
Проверяем версию:
```
brew —version
```
Устанавливаем gdb:
```
brew install gdb
```
Проверяем
```
gdb —version
```

Generate a certificate

If it was as simple as installing gdb, I wouldn't have made this guide! If you attempt to debug a file now, you will get errors and unexpected behavior because the Darwin kernel does not allow gdb to control another process without having special rights. Fortunately the rest of this guide walks you through all of the steps you'll need in order to use gdb effectively on your Mac.
In order to give gdb the permissions it needs, you'll need to generate a self-signed certificate. Here are the steps:

1. Launch Keychain Access application: Applications > Utilities > Keychain Access.
2. From the Keychains list on the left, right-click on the System item and select Unlock Keychain "System".
3. From the toolbar (сверху), go to Keychain Access > Certificate Assistant > Create a Certificate.
4. Choose a name (например, gdb-cert).
5. Set Identity Type to Self Signed Root.
6. Set Certificate Type to Code Signing.
7. Check the Let me override defaults checkbox.
8. (Пройти через все экраны, ничего не меняя. На последнем экране установить Keychain to System. Кликнуть Create button) At this point, you can go on with the

installation process until you get the Specify a Location For The Certificate dialogue box. Here you need to set Keychain to System. Finally, you can click on the Create button.

9. After these steps, you can see the new certificate under System keychains. From the contextual menu of the newly created certificate (right-click on it) select the Get info option. In the dialogue box, expand the Trust item and set Code signing to Always Trust.
10. Then, from the Keychains list on the left, right-click on the System item and select Lock Keychain "System".
11. Finally, reboot your system.

Sign certificate for gdb

Now you'll need to sign the certificate.

- открыть терминал
- Выйти в корень
  cd
- Путь будет: **/Users/dmitry**

- Create a file called **gdb-entitlement.xml**. This allows macOS to trust gdb operations for debugging. Type the following into the xml file you just created, and save it:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>com.apple.security.cs.allow-jit</key>
    <true/>
    <key>com.apple.security.cs.allow-unsigned-executable-memory</key>
    <true/>
    <key>com.apple.security.cs.allow-dyld-environment-variables</key>
    <true/>
    <key>com.apple.security.cs.disable-library-validation</key>
    <true/>
    <key>com.apple.security.cs.disable-executable-page-protection</key>
    <true/>
    <key>com.apple.security.cs.debugger</key>
    <true/>
    <key>com.apple.security.get-task-allow</key>
    <true/>
</dict>
</plist>
```

Now back in the Terminal app, navigate to the directory where you saved the xml file.
*note. <gdbPath> in the following command is your path to gdb. It will be something like
/usr/local/Cellar/gdb/version/bin/gdb
or
/usr/local/bin/gdb.
You can find out for sure by typing the following in the terminal:

```
which gdb
```

*** У меня установлен в /usr/local/bin/gdb

Replace <gdbPath> with your path to gdb.

```
codesign --entitlements gdb-entitlement.xml -fs gdb-cert /usr/local/bin/gdb
```

where gdb-cert is the name of the certificate you created earlier, and gdbPath is the full path to your gdb binary file.

Create a gdb command file

For this step, do one of the following.
Either:

- In the home directory, create a new file called **.gdbinit**. Write the following command into it and save:

```
set startup-with-shell off
```

or

- Type the following into the Terminal:

```
echo "set startup-with-shell off" >> ~/.gdbinit
```

Generating builds

We're almost there! If you tried debugging with gdb now, you would likely receive a "No symbol table is loaded" error. In order to solve this, you'll need to compile programs with the -ggdb option, as in the following example:

```
gcc hello_world.c -o hello_world -ggdb.
```

At this point you should be able to successfully debug your programs on Mac using gdb. Do a test by running (for example):

```
gdb hello_world
```

///ДАЛЬШЕ НЕ ДЕЛАТЬ ///

and then from the gdb prompt, try creating a breakpoint:

```
(gdb)break main
```

You should see something like the following:

```
Breakpoint 1 at 0x100000de5: file hello_word.c, line 15.
```

I hope you're seeing the results you expected! Leave a comment and let me know how it goes!

Дальше: https://habr.com/ru/post/491534/