# Abanu Project Documentation

Dec 21, 2019

# Contents

CHAPTER 1

# Building Abanu

Abanu can be build platform independent. However, we support primary an unix build environment. For Windows Users, we offer a Step by Step guide.

## 1.1 Building on Linux

You need some packages:

- `git` version control system
- `build-essential` Build tools
- `wget` for downloading additional ressorces
- `nasm` Assembler
- `mtools` Creating FAT disk images
- `grub-*` for creating boot images with bootloader. It will not affect your current system. We need the binaries for the disk creation.
- `xorriso` Disk creation

You can install them all via:

```
sudo apt-get install -y --no-install-recommends git get nasm qemu-system-x86 mtools␣
↪xorriso grub-common grub-pc-bin grub-efi-amd64-bin grub-efi-ia32-bin
```

If you want to debug Abanu, you also need `gdb`.

Download and build:

```
git clone --recursive https://github.com/abanu-org/abanu.git
cd abanu
make
```

To run the kernel, just execute `./abctl run qemu x86-grub-vbe`. To debug the kernel, run `./abctl debug qemu-kernel`

## 1.2 Building on Windows

### 1.2.1 The Quick way

If you want only start run Abanu, just get the sources, open Abanu.sln in Visual Studio, compile whole solution launch the default project (`Abanu.Tools.Build`). However, this is only a shortcut. If you want debug Abanu, you may need the following steps.

### 1.2.2 Install the Windows Subsystem for Linux

Before installing any Linux distros for WSL, you must ensure that the "Windows Subsystem for Linux" optional feature is enabled:

1. Open PowerShell as Administrator and run:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

2. Restart your computer when prompted.

Visit Microsoft App Store, and Download the App Debian

Populate the Debian System with Packages:

```
# run as root
apt-get update && apt-get install -y wget
wget -qO- https://raw.githubusercontent.com/abanu-org/abanu/master/build/debian/
↪install | bash -s
```

This will take a while. After that, you have a fully featured build environment.

### 1.2.3 Enabling Graphical Unix Applications

To launch graphical applications like Geany or Qalculate, you need XLaunch / VcXsrv Windows X Server. We can do this all via command line:

In Windows, install the packet manager chocolatey. Open a PowerShell with Administrator rights and run:

```
Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.
↪WebClient).DownloadString('https://chocolatey.org/install.ps1'))
```

Install the package VcXsrv:

```
choco install -y vcxsrv
```

You can launch now `XLaunch` from Start menu. Simply follow the wizard without making changes.

Open the Debian app and finish the install instructions. When command prompt is ready, start a dedicated Terminal:

```
export DISPLAY=:0 && xfce4-terminal &
```

Now you can run unix applications even with graphical user interface.

### 1.2.4 Additional Tools for Windows:

```
choco install -y git
```

### 1.2.5 Share project directory

Because Visual Studio cannot open projects via `\\$wsl`, you have to place the files on the windows drive and link that folder to the WSL home folder. Run this commands in a WSL/Debian bash shell:

```
# specify root folder for projects.
WINPROJDIR=$(cmd.exe /C "echo|set /p=%USERPROFILE%")/Documents/abanu-org
# normalize windows path
WINPROJDIR=$(wslpath -w $(wslpath -u $WINPROJDIR))
# create the windows project root
cmd.exe /C mkdir $WINPROJDIR
# create symbolic link
ln -s $(wslpath -u $WINPROJDIR) ~/
# Switch to new directory
cd ~/abanu-org
```

Now `/home/<user>/abanu-org` and `C:\Users\<user>\Documents\abanu-org` points to the same directory.

### 1.2.6 Download and build Abanu

```
git clone --recursive https://github.com/abanu-org/abanu.git
cd abanu
./abctl configure packages
./abctl build all
```

Now you can run abanu in qemu:

```
./abctl debug qemu-kernel
```

# CHAPTER 2

# TODO

**Compiler:**

- STACK_TOP configurable
- Patch for ProgramHeader [DONE]
- Insert Comment for explanation why needing Running InitializeAssembly() manually [DONE]
- Adding more DWARF sections to debug local variables.

**Kernel:**

- Check Permission of all SysCalls, like GetPhysicalMemory
- Don't run all apps with full IOCPL permissions
- Add interprocess synchronization mechanisms like mutex and semaphore.
- no kernel panic if user apps is crashing
- Switch from Text Mode to Graphics Mode without boot loader, if booted in text mode.
- Add x64 support

**Services:**

- avoid using "unsafe" keyword.
- Using more ref structs.
- Finalize File-System Interface
- Free Handles on User app quit
- Ext2-Driver
- Network stack
- Basic USB-Support
- Move ConsoleHost to central Service
- Multiplexing ConsoleHost for multiple clients / allow multiple virtual consoles

**Libs:**

Implement Memory-/Allocation-friendly Dictionary "KDictionary".
Porting "newlib" with basic SysCalls.

**Apps:**

Simple Command Line Interpreter
Porting true, false, cat, readline and echo, as native c apps for proof of concept unix app tests, based on newlib.

**Tools:**

Adding automated tests
Optimize Argument parsing of abctl to allow optional parameters (–name value) and bulk actions (build a,b,c)
Attaching “Networkdisk” to all kinds of qemu runs.
Add helper function to abctl to start/stop HostCommunication.exe

CHAPTER 3

# About this documentation

This document focuses on style-guide and a short reference. It is a kind of coding standards applied to documentation files. It is not about documentation content.

Table 1: Links to Documentation

| | |
|---|---|
| HTML, Online | https://docs.abanu.org/en/latest |
| PDF | https://readthedocs.org/projects/abanu/downloads/pdf/latest |
| HTML, as Zip | https://readthedocs.org/projects/abanu/downloads/htmlzip/latest |

## 3.1 RestructuredText with Sphinx directives

This documentation uses Python-sphinx[1], which itself uses reStructuredText[2] syntax.

## 3.2 Filenames

Use only lowercase alphanumeric characters and − (minus) symbol.

Suffix filenames with the `.rst` extension, so GitHub can render them.

## 3.3 Whitespaces

### 3.3.1 Indentation

Indent with 2 spaces.

Except:

[1] http://sphinx.pocoo.org/
[2] http://docutils.sourceforge.net/rst.html

- `toctree` directive requires a 3 spaces indentation.

### 3.3.2 Blank lines

Two blank lines before overlined sections, i.e. before H1 and H2. One blank line before other sections. See *Headings* for an example.

One blank line to separate directives.

```
Some text before.

.. note::

  Some note.
```

Exception: directives can be written without blank lines if they are only one line long.

```
.. note:: A short note.
```

### 3.3.3 Line length

Technically, there's no limitation. But if possible, limit all lines to a maximum of 120 characters.

## 3.4 Headings

- Please stick to this order of heading adornments:
  1. = with overline for document title:

     ```
     ==============
     Document title
     ==============
     ```

  2. = for chapters:

     ```
     Chapters
     ========
     ```

  3. – for sections:

     ```
     Section
     -------
     ```

  4. ~ for subsections:

     ```
     Subsection
     ~~~~~~~~~~
     ```

If you need more than heading level 4 (i.e. H5 or H6), then you should consider creating a new document.

There should be only one H1 in a document.

**Note:** See also Sphinx's documentation about sections[3].

## 3.5 Code blocks and text boxes

Use the `code-block` directive **and** specify the programming language. As an example:

```
.. code-block:: python

  import this
```

Text boxes:

```
.. note::

   Note (blue box). possible values: attention, caution, danger, error, hint,␣
↪important, note, tip, warning, admonition.
   Every type has its own color.
```

will look like:

**Note:** Note (blue box). possible values: attention, caution, danger, error, hint, important, note, tip, warning, admonition. Every type has its own color.

## 3.6 Links and references

Use links and references footnotes with the `target-notes` directive. As an example:

```
==============
Some document
==============

Link without Reference: `Example <http://www.example.com>`__

Some text which includes links to `Example website`_ and many other links.

`Example website`_ can be referenced multiple times.

(... document content...)

And at the end of the document...

References
----------

.. target-notes::

.. _`Example website`: http://www.example.com/
```

[3] http://sphinx.pocoo.org/rest.html#sections

## 3.7 Tables

Table as CSV

```
.. csv-table:: Title of CSV table
  :header: "Column 1", "Column 2", "Column 3"

  "Sample Row 1", Cell, Cell
  "Sample Row 2", Cell, "Cell with multiple Words"
```

You can skip quotes, of cell content contains only a single word

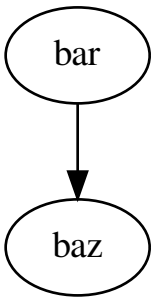
Table as flat list

```
.. list-table:: Title of table as flat list
  :header-rows: 1

  * - Column 1
    - Column 2
    - Column 3
  * - Row 1
    - Cell
    - Cell
  * - Row 2
    - Cell
    - Cell
```

`:header-rows:` defines the number of header rows. Skip this line, if you do not need a header.

## 3.8 Diagrams

```
.. graphviz::

   digraph foo {
      "bar" -> "baz";
   }
```



## 3.9 Troubleshooting

### Why is my document not linked within the table of contents?

- put the filename into index.rst. Omit the `.rst` extension.
- The document requires at least one section. Section names are the label that are used for the table of content.

### After committing, the documentation will not update

- The build process of the documentation takes round about 1-2 minutes.
- You can check the status here: Builds
- Check the status if the build fails

### Documentation is updating, but some content is missing or malformed

- There might be some parsing errors or warnings. Go to Builds and click on `Raw view`. Check the build output for warnings and fix them.

## 3.10 References

- https://sphinx-rtd-theme.readthedocs.io/en/latest/demo/demo.html
- http://www.ericholscher.com/blog/2016/jul/1/sphinx-and-rtd-for-writers/