2026 CAE Community Symposium

Delivering Cyber Education Content in Browser-Based Virtual Machines (BBVMs)

PennState College of Information Sciences and Technology

Oregon State University

Nick Giacobe

Associate Teaching Professor, Penn State University

and

Matt Ruff

Assistant Professor of Practice, Oregon State University

Problem Statement

2025 CAE Community Symposium

Students need

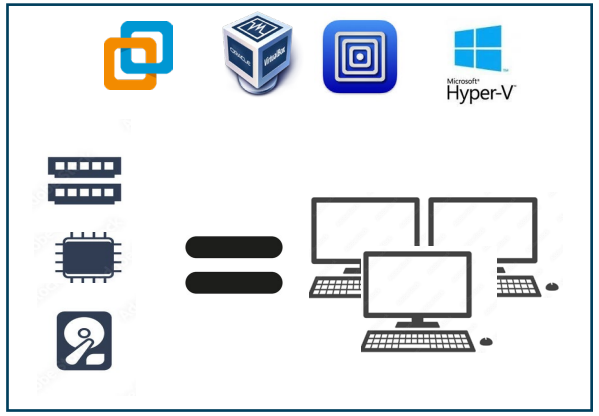
- **Easy to Access**
- Structured activity, with a goal, and a right answer
- Multiple, repeatable attempts
- Engagement – a.k.a “fun”

Organizations - Universities

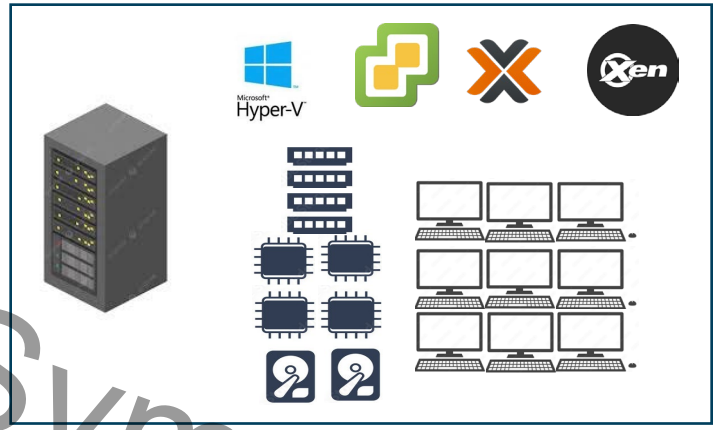
- **No overhead** (or very little overhead)
- Prevent cheating
- Automatic assessment of the right answer

What BBVMs are not…

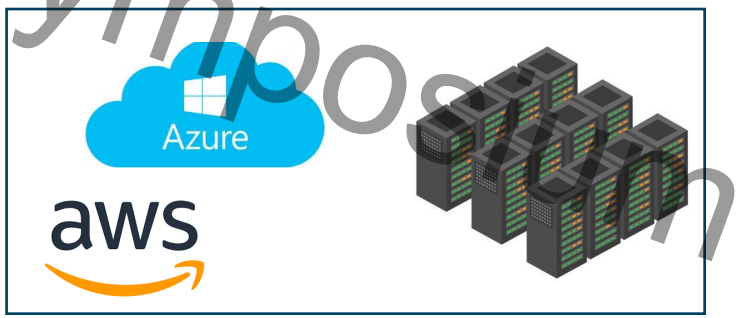
Computer + Virtualization =
Desktop (Type II)
Virtualization



Hypervisor + Multiple VMs =
Bare Metal (Type I)
Virtualization



Hypervisor + High Speed
Internet = **Cloud Computing**



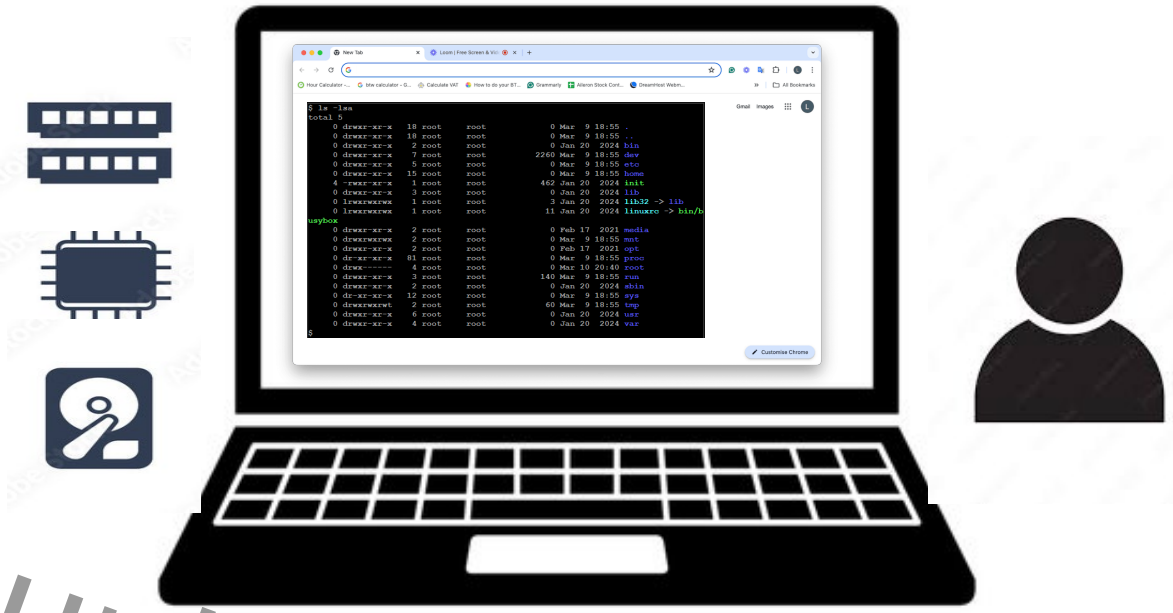
2025 CAE Community Symposium

What are BBVMs?

VM + Web Assembly =
Browser-Based Virtual Machine (BBVM)

Apache or nginx

+ v86 Project



... and then the VM runs inside the browser's Javascript VM

2025 CAE Community Symposium

Some of the History of BBVMs

- https://bellard.org/jslinux/
  - Alpine Linux, Buildroot, Fedora 33, Win2000, FreeDOS
- http://copy.sh/v86
  - Arch, DSL, Buildroot, Win 1.01/3.1/95/98/2000, MS-DOS 6.22, FreeDOS, Open BSD, FreeBSD, Several others

2025 CAE Community Symposium

2025 CAE Community Symposium

Serving BBVMs via your web server with v86

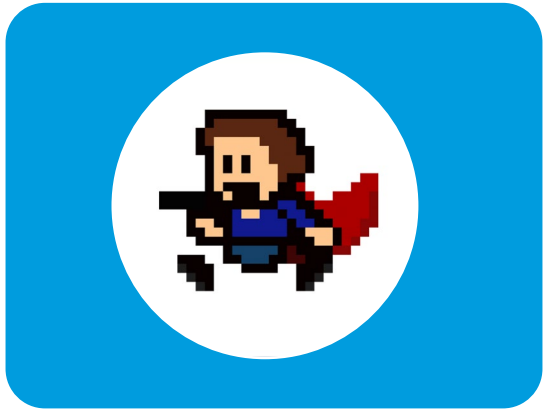
PennState
College of Information Sciences and Technology

Oregon State University

Serve BBVMs with v86

- Deliver BBVMs to students/participants
- Leverages Copy.sh's v86 project
- WASM, NodeJS, Rust, etc.
- Deliver via Apache or NGINX
- Host on AWS, Azure, or locally

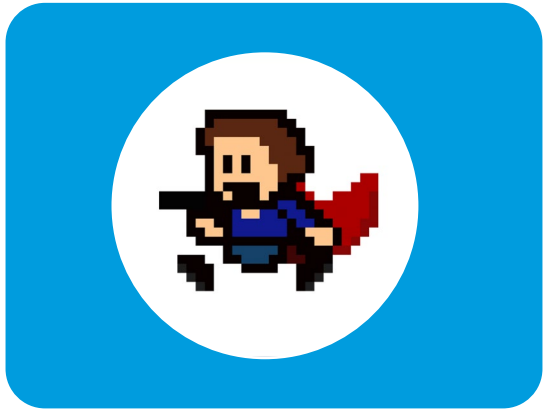
Running the v86 Project on your web server



2025 CAE Community Symposium

- The v86 project gives the ability to serve different operating system .iso files as javascript-based operating systems.
- v86 is the emulator that these operating systems run on.
- You can run v86 on a variety of web servers. These instructions provide the ability to run the project on a python-based simplified web server that has been executed on an Ubuntu operating system (Ubuntu 22.04 LTS server and/or desktop)

Setup v86 project on your Linux machine



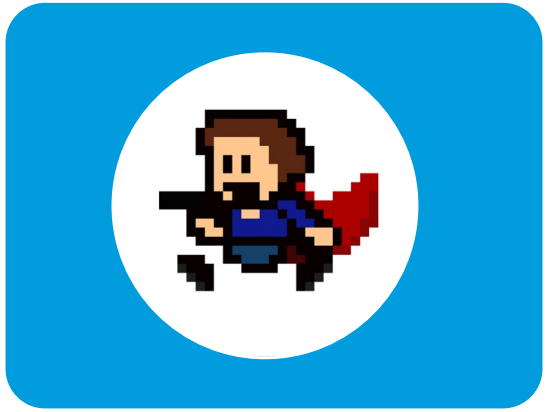
2025 CAF Community Symposium

In the directory where browser-vm is, get and expand the following files from the v86 release directory

```
wget https://github.com/copy/v86/archive/refs/tags/latest.tar.gz
tar -xvzf latest.tar.gz
```

Note: I did this to my local home directory, where I have full permissions. This becomes important later. Resist the urge to put this file in your /var/www/html directory. You need to build this first where you have permissions, then MOVE the results to the /var/www/html directory later.

Get the other 3 files from the releases directory



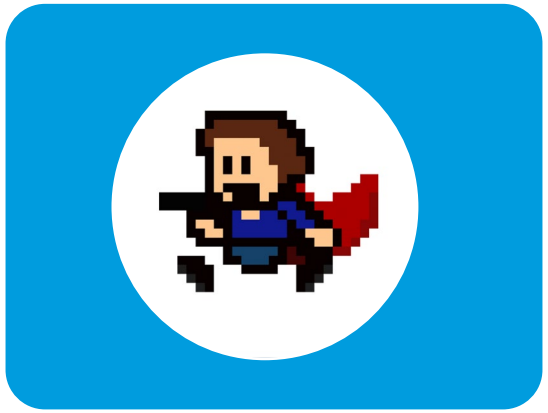
- wget the following 3 files from the releases directory:
  - libv86.js, v86-fallback.wasm, v86.wasm

```
wget https://github.com/copy/v86/releases/download/latest/libv86.js
wget https://github.com/copy/v86/releases/download/latest/v86-fallback.wasm
wget https://github.com/copy/v86/releases/download/latest/v86.wasm
```

Put these all in the v86-latest folder you created when you expanded the tar.gz file. (mine was ~/v86-latest)

2025 CAE Community Symposium

Install dependencies



- Install make

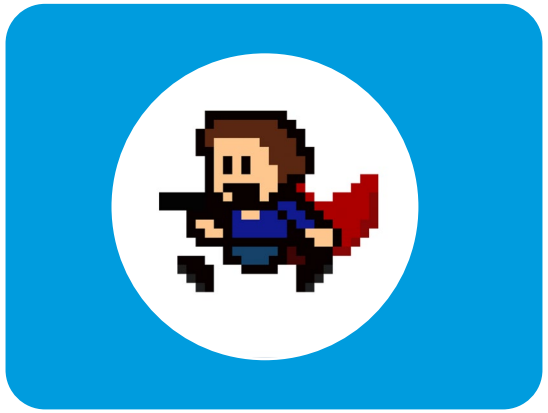
```
sudo apt install make
```

- Install other dependencies

```
sudo apt install openjdk-17-jdk build-essential libc6-i386
clang nasm gdb qemu-system gcc rustfmt curl -y
```

2025 CHASE Community Symposium

Other fussier dependencies



- nodejs

  ```
  sudo apt install nodejs
  ```

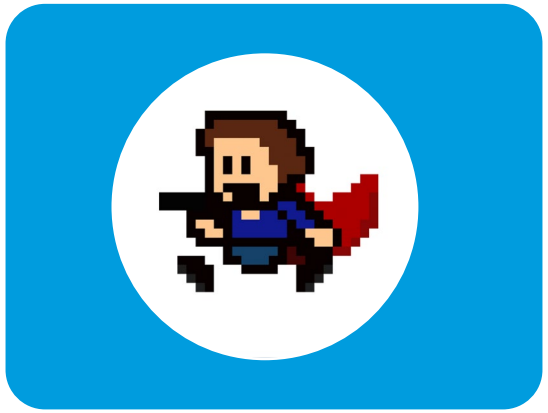
- rust

  ```
  sudo curl https://sh.rustup.rs -sSf | sh
  ```

  - Note: here you might have to uninstall rust if it's already installed. You cannot have rustup and rust both installed
  - Setup rust (with rustup) with the wasm32-unknown-unknown target profile
    ```
    rustup target add wasm32-unknown-unknown
    ```
  - and set the cargo environment variable location
    ```
    source "$HOME/.cargo/env"
    ```

2020 CAE Community Symposium

Get the .iso files you need to run v86 as-is

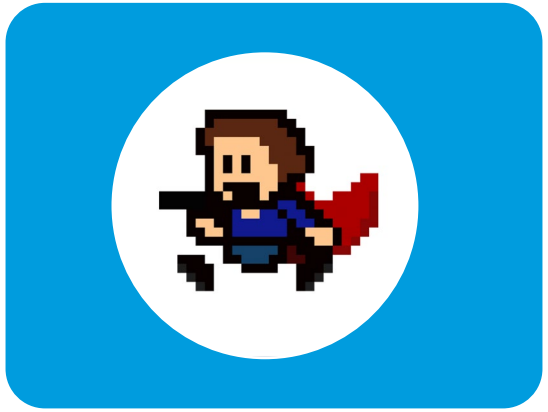


```
cd ~/v86-latest
git clone https://github.com/copy/images
```

- This will populate your /v86-latest/images directory with several .iso files
- You didn't have to do all of this the "hard way" just to run the samples from v86, but you might want these for other purposes…

2025 C-AC Community Symposium

Get the .iso files you need to run v86 as-is



2025 CAE Community Symposium

copy / images Public

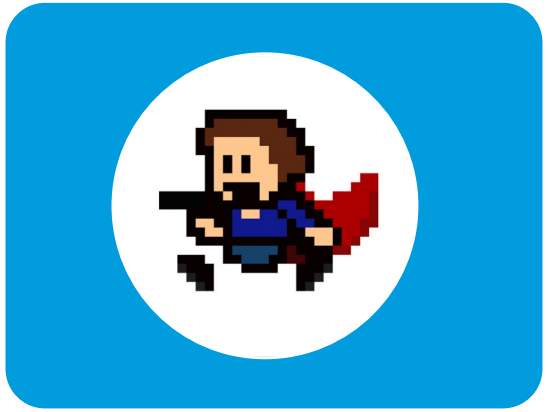
Code | Issues 1 | Pull requests | Actions | Projects | Wiki | Security

master | 1 Branch | 0 Tags | Go to file

266-750Balloons and copy Add floppy bird as a test image. db92d8f · 4 years ago

Readme.md	Add floppy bird as a test image.
freedos722.img	fuerst commit
kolibri.img	Improved images
linux.iso	Improved images
msdos.img	fuerst commit
openbsd.img	better working openbsd image
os8.dsk	fuerst commit
windows101.img	fuerst commit

Now run the .py web server to test everything from the v86-latest directory



2025 CAE Community Symposium

- Compile the code and run the web server for the first time

  ```
  make
  make all
  ```

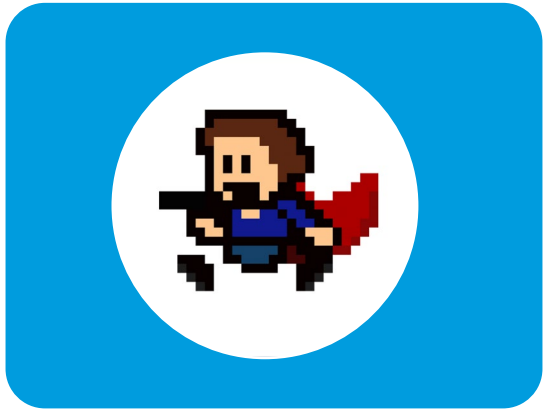
- Optional – Test run with the Python Web Server

  ```
  make run
  ```

  - Open your browser to 0.0.0.0:8000, which should give you the v86 project page.

- Now, consider making a snapshot at this point

Move the v86-latest folder to the web server's directory



2025 CAF Community Symposium

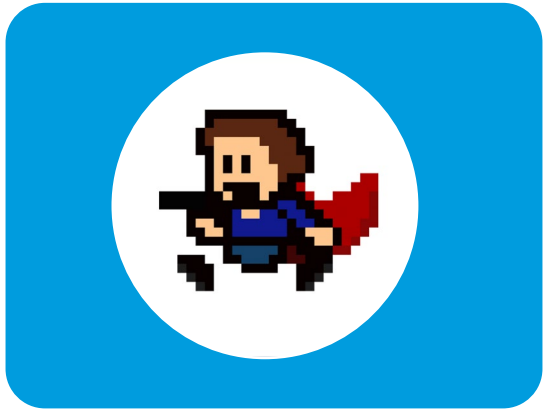
- For nginx or apache – move the whole v86-latest folder to /var/www/html

```
cd ~
```

```
sudo mv v86-latest /var/www/html
```

- Or wherever you want it to be
- Alternatively, change your configuration file of your web browser to serve the files from the directory where v86 was installed

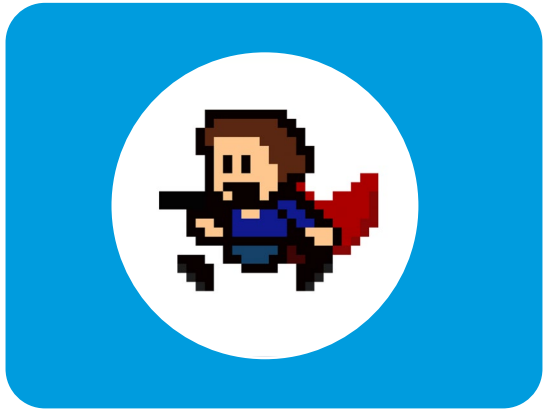
Serving your own .iso files



- This is why you're doing this, right?
- Save the .iso files in the /images folder
- Modify the web page the files in one of two places
  - /v86-latest/examples directory – modify one of the files (like the basic.html) to reference your .iso
  - In /v86-latest/build -- Change the v86_all.js file

2025 CAE Community Symposium

Using the /examples directory



2025 CAE

- There are several examples in the /examples directory
- Most follow a simple HTML file structure
- Modify one (like basic.html) to point to your .iso file

```
var emulator = window.emulator = new V86Starter({
        wasm_path: "../build/v86.wasm",
        memory_size: 32 * 1024 * 1024,
        vga_memory_size: 2 * 1024 * 1024,
        screen_container: document.getElementById("screen_container"),
        bios: {
            url: "../bios/seabios.bin",
        },
        vga_bios: {
            url: "../bios/vgabios.bin",
        },
        cdrom: {
            url: "../images/linux.iso",
        },
        autostart: true,
    });
```

Whatever Your .iso File is

Symposium

Baseline OS Setup

- Ubuntu Linux 22.04 LTS – Desktop
- Other Packages
- Update the package manager

```
sudo apt update
sudo apt upgrade
```

- Other dependent packages

```
sudo apt install git
sudo snap install docker
```

- Then, likely restart

2025 CAE Community Symposium

Note: Use snap to install Docker, not apt

- There are two package managers on Ubuntu
  - APT
    - The old one
  - SNAP
    - The new one

- Docker should be installed with SNAP

```
sudo snap install docker -y
```

2025 CAE Community Symposium

BrowserVM to create a buildroot .iso file

2025 FAE Community Symposium

```
git clone https://github.com/humphd/browser-vm.git
cd browser-vm
sudo docker build -t buildroot .
sudo docker run \
    --rm \
    --name build-v86 \
    -v $PWD/dist:/build \
    -v $PWD/buildroot-v86/:/buildroot-v86 \
    -ti \
    --entrypoint "bash" \ buildroot
```

Drops you into root into that docker container you just built

These are the steps from humphd's browser-vm package. You run these inside the docker container you just set up


```
$ make BR2_EXTERNAL=/buildroot-v86 v86_defconfig
$ make menuconfig
...
$ make savedefconfig
$ make linux-menuconfig
...
$ make linux-savedefconfig
$ make
```

Complete the text-based user interface menu choices. There are two of these.
Accept all of the defaults your first time doing this.

PS: If you get an error after "make linux-menuconfig" saying some licenses directory and files were not found, don't panic. You can ignore those and continue with next steps. These does not affect building ISO. Of course, there will be subsequent errors w.r.t. "license" but ignore those as well.

2018 CAE Community Symposium

Now, let's see what you've created, and hang on to the virtual machine for the future..

2026 CAE Community Symposium

- Look in the host OS, in the /browser-vm/dist folder
- You should see
  - v86-linux.iso ← this is an iso of a bootable buildroot image, which you can port over to your /v86 project folder and serve up.
- What happens if you re-start the container?

```
./browser-vm/install.sh
```

or

```
sudo docker exec buildroot bash
```

- This re-starts the whole re-compilation of the buildroot vm, from scratch. Ugh.

“make” our .iso file again

- From the container’s root user prompt (#), issue

```
make
```

- We can re-run the last command to “make” another iteration of v86-linux.iso – but it only makes sense to do this if you want other files in your file system, so go put it in /browser-vm/buildroot-v86/board/v86/rootfs_overlay
- Modify the the overlay directory, then “make” again to spit out another .iso file (It'll overwrite the existing .iso)

2025 CAE Community Symposium

The overlay directory doesn't have much in it

2025 CAE Community Symposium

...but you can add things that will get merged with the file system when you "make" it. So add things to /browser-vm/buildroot-v86/board/v86/rootfs_overlay

Create folders that don't exist yet in the overlay directory like

- /root
  - And .profile in the root folder, so root does stuff when it logs in
  - And README.txt in the root folder
  - And other files you might need
- /home
  - For each user account that might get created

- modify the file /etc/inittab
  - Consider changing this line (which will require login on startup)

    `console::respawn:/sbin/getty -L  console 0 vt100 # GENERIC_SERIAL`

    To this to avoid logging on and just run the shell as root

    `::respawn:~/bin/sh`

    Or to this to automatically log on as root

    `::respawn:~/bin/login -f root`

Thank You!

… And Now Your Questions

2025 CAE Community Symposium

PennState College of Information Sciences and Technology

Oregon State University

Nick Giacobe

Associate Teaching Professor, Penn State University

and

Matt Ruff

Assistant Professor of Practice, Oregon State University