gem5 Full System Simulation

A presentation by

Maryam Babaie

UCDAVIS COMPUTER SCIENCE

DArchR DAVIS ARCHITECTURE RESEARCH

OOO Action Item ☺

Launch codespace and run the following commands:

```
cd gem5
```

```
scons build/RISCV/gem5.opt -j14
```

Table of Contents

Intro. to gem5 Full System Mode

What is full-system simulation?

Full-system Simulation (FS) encompasses the entire computer system:

- the processor cores
- peripheral devices
- memories
- network connections
- the complete software stack
  - device drivers
  - operating systems
  - application programs

gem5 in Full System Mode

- In FS mode, gem5 simulates the entire hardware system, from the CPU to the I/O devices.
- It helps gem5 execute binaries with no modifications.
- gem5's full system mode enables us to investigate the impacts of the operating system.

What gem5 needs for FS?

core of an OS

A descriptive file of the hard drive, the non-volatile memory to store data, app, bootloader and OS, etc

gem5 Binary Config file

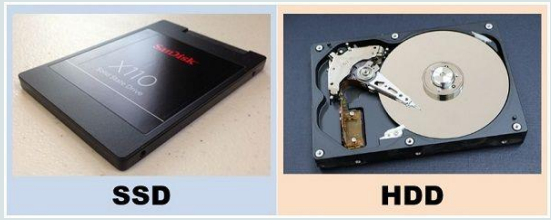
Kernel Binary

Disk Image

Workloads Binaries

x86-npb-benchmarks.py

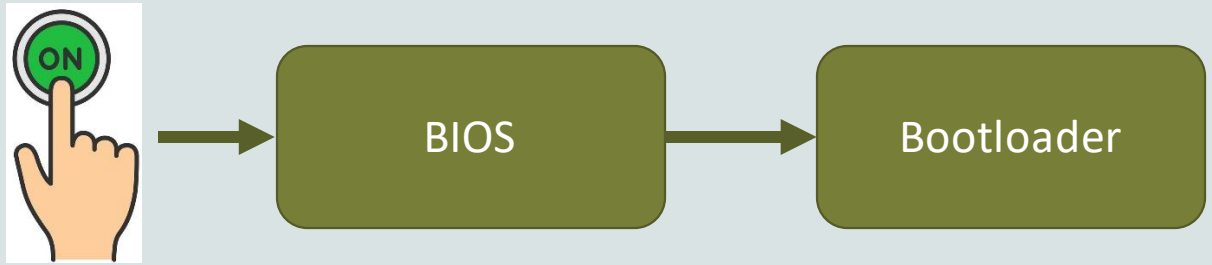
gem5/build/**X86**/gem5.opt
gem5-**arm**



gem5/build/X86/gem5.opt x86-npb-benchmarks.py --PATH-TO-KERNEL-BIN --PATH-TO-DISK-IMG --PATH-TO-WKL-BIN

Basics of Booting up a System

Booting Steps



BIOS (basic input/output system) software is stored on a non-volatile ROM chip on the motherboard.

BIOS perform hardware initialization during the booting process.

Bootloader

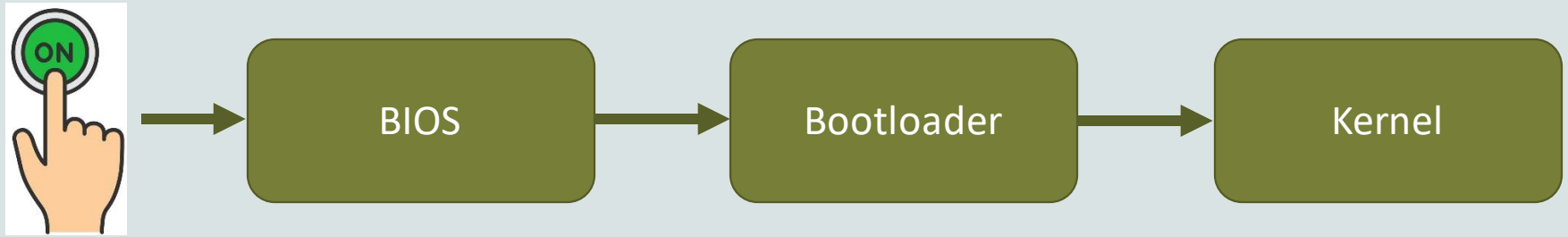
OS must be loaded into the working memory once the computer is starting up.

This is the job of a bootloader!

Immediately after a device starts, a bootloader is launched by a bootable medium like a hard drive.

The bootable medium receives information from the computer's firmware (e.g. BIOS) about where the bootloader is.

Booting Steps



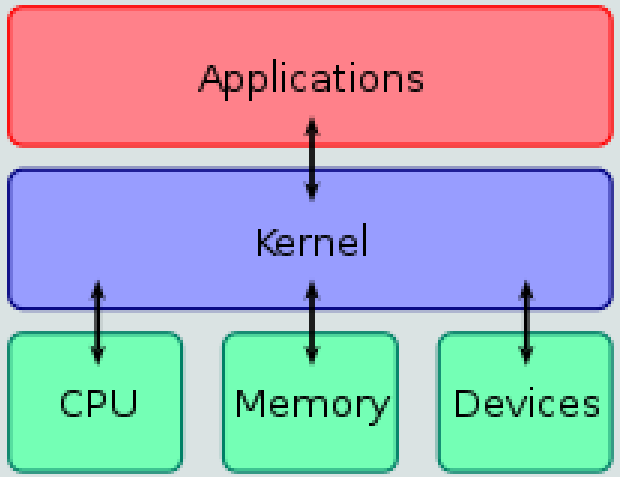
Kernel

It's the core of an OS and generally has complete control over everything in the system.

It resides in the main memory.

Facilitates interactions between hardware and software components.

The most well-known one is the Linux kernel.



Device Tree Binary (DTB)

A device tree is a data structure describing the hardware components of a particular computer so that the operating system's kernel can use and manage those components, including:

- the CPUs
- the memory
- the buses
- the integrated peripherals.

Booting Steps

BIOS → Bootloader → Kernel **(linux)** → Systemd

It's a system and services manager, the **glue** between kernel and the application/user

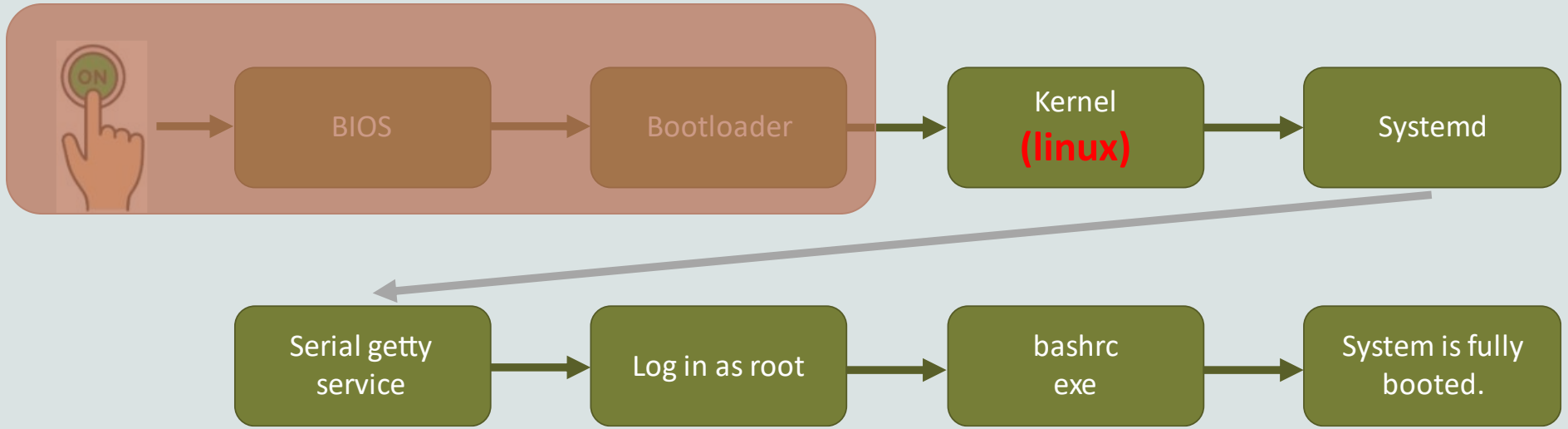
Systemd → Serial getty service → Log in as root → bashrc exe → System is fully booted.

The bashrc file is a script file that's executed when a user logs in. The file contains a series of configs for the terminal session.

gem5

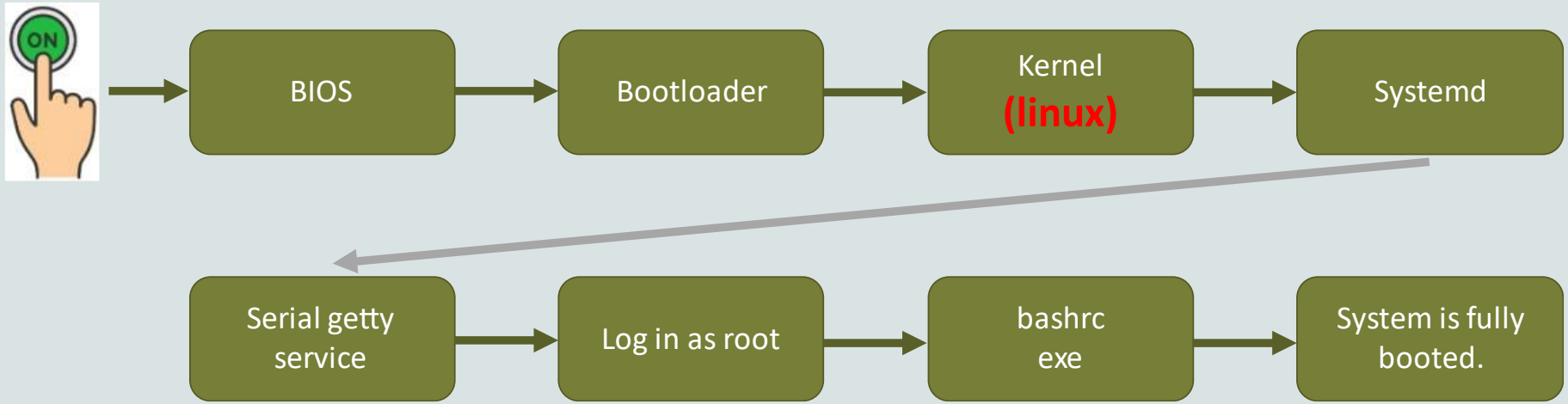
Booting Steps: gem5

1. gem5 does not simulate the BIOS/Bootloader. It directly loads kernel into the memory and continues the booting.
2. Running “systemd” is optional, instead other user-defined init app/services can be used.



Why Disk Image Matters?

Once we are creating a disk image for our FS simulation, we need to make sure that proper files and programs (in a matching ISA to the target) are loaded into the disk image, enabling this chart to happen once you start the simulation.



Creating Disk Image by Packer & QEMU

Disk Image

- Disk image is a file containing the contents and structure of a data storage device, such as a hard disk drive.
- A disk image is usually made by creating a sector-by-sector copy of the source medium, so it replicates the structure and contents of a storage device independent of the file system.
- The file format may be an open standard, such as the ISO image format for optical disc images.
- The size of a disk image can be large because it contains the contents of an entire disk.

How to create your disk image?

- Using gem5 utils to create a disk image
- Using QEMU to create a disk image
- Using Packer to create a disk image
  - Pros: it automates 2
  - Cons: can only be used for X86 for automation, for ARM and RiscV you may want to do it with QEMU

Using Packer

- We use Packer with QEMU to automate the process of disk creation.
- QEMU is responsible for setting up a virtual machine and all interactions with the disk image during the building process.
  - installing Ubuntu Server to the disk image
  - copying files from host machine to the disk image
  - running scripts on the disk image after Ubuntu is installed.
- However, we will not use QEMU directly. Packer provides a simpler way to interact with QEMU using a **.json script**.

Creating Disk Image with Packer

We'll look into NAS parallel benchmarks (NPB) from gem5 resources.
NPB consists of a set of high performance computing (HPC) workloads.

$ build.sh ×

gem5-resources > src > npb > disk-image > $ build.sh

Hoa Nguyen, 19 months ago | 1 author (Hoa Nguyen)

```
1  PACKER_VERSION="1.7.8"
2
3  if [ ! -f ./packer ]; then
4      wget https://releases.hashicorp.com/packer/${PACKER_VERSION}/packer_${PACKER_VERSION}_linux_amd64.zip;
5      unzip packer_${PACKER_VERSION}_linux_amd64.zip;
6      rm packer_${PACKER_VERSION}_linux_amd64.zip;
7  fi
8
9  ./packer validate npb/npb.json
10 ./packer build npb/npb.json
```

Packer Script: npb.json

```
"builders":
[
    {
        "type": "qemu",
        "format": "raw",
        "accelerator": "kvm",
        "boot_command":
        [
            "{{ user `boot_command_prefix` }}",
            "debian-installer={{ user `locale` }} auto locale={{ u
            "file=/floppy/{{ user `preseed` }} ",
            "fb=false debconf/frontend=noninteractive ",
            "hostname={{ user `hostname` }} ",
            "/install/vmlinuz noapic ",
            "initrd=/install/initrd.gz ",
            "keyboard-configuration/modelcode=SKIP keyboard-config
            "keyboard-configuration/variant=USA console-setup/ask_
            "passwd/user-fullname={{ user `ssh_fullname` }} ",
            "passwd/user-password={{ user `ssh_password` }} ",
            "passwd/user-password-again={{ user `ssh_password` }}
            "passwd/username={{ user `ssh_username` }} ",
            "-- <enter>"
        ],
        "cpus": "{{ user `vm_cpus`}}",
```

```
"provisioners":
[
    {
        "type": "file",
        "source": "../gem5/util/m5/build/x86/out/m5",
        "destination": "/home/gem5/"
    },
    {
        "type": "file",
        "source": "shared/serial-getty@.service",
        "destination": "/home/gem5/"
    },
    {
        "type": "file",
        "source": "npb/runscript.sh",
        "destination": "/home/gem5/"
    },
    {
        "type": "file",
        "source": "npb/npb-hooks/NPB3.3.1/NPB3.3-OMP",
        "destination": "/home/gem5/"
    },
    {
        "type": "shell",
        "execute_command": "echo '{{ user `ssh_password` }}' | {{.Vars}}
        "scripts":
        [
            "npb/post-installation.sh",
            "npb/npb-install.sh"
        ]
    }
],
```

```
"variables":
{
    "boot_command_prefix": "<enter><wait><f6
    "desktop": "false",
    "image_size": "12000",
    "headless": "true",
    "iso_checksum": "34416ff83179728d54583bf
    "iso_checksum_type": "md5",
    "iso_name": "ubuntu-18.04.2-server-amd64
    "iso_url": "http://old-releases.ubuntu.c
    "locale": "en_US",
    "preseed" : "preseed.cfg",
    "hostname": "gem5",
    "ssh_fullname": "gem5",
    "ssh_password": "12345",
    "ssh_username": "gem5",
    "vm_cpus": "4",
    "vm_memory": "8192",
    "image_name": "npb"
```

post-installation.sh

$ post-installation.sh ×

gem5-resources > src > npb > disk-image > npb > $ post-installation.sh

Ayaz Akram, 2 years ago | 1 author (Ayaz Akram)

```
1  #!/bin/bash
2
3  # Copyright (c) 2020 The Regents of the University of California.
4  # SPDX-License-Identifier: BSD 3-Clause
5
6  echo 'Post Installation Started'
7
8  mv /home/gem5/serial-getty@.service /lib/systemd/system/
9
10 mv /home/gem5/m5 /sbin
11 ln -s /sbin/m5 /sbin/gem5
12
13 # copy and run outside (host) script after booting
14 cat /home/gem5/runscript.sh >> /root/.bashrc
15
16 echo 'Post Installation Done'
```

npb-install.sh

$ npb-install.sh ×

gem5-resources > src > npb > disk-image > npb > $ npb-install.sh

Ayaz Akram, 2 years ago | 1 author (Ayaz Akram)

```
1   #!/bin/sh
2
3   # Copyright (c) 2020 The Regents of the University of California.
4   # SPDX-License-Identifier: BSD 3-Clause
5
6   # install build-essential (gcc and g++ included) and gfortran
7
8   #Compile NPB
9
10  echo "12345" | sudo apt-get install build-essential gfortran
11
12  cd /home/gem5/NPB3.3-OMP/
13
14  mkdir bin
15
16  make suite HOOKS=1
```

runscript.sh

$ runscript.sh

gem5-resources > src > npb > disk-image > npb > $

Ayaz Akram, 2 years ago | 1 author (Ayaz Akram)

```
1   #!/bin/sh
2
3   # Copyright (c) 2020 The Regents of
4   # SPDX-License-Identifier: BSD 3-Cl
5
6   m5 readfile > script.sh
7   if [ -s script.sh ]; then
8       # if the file is not empty, exe
9       chmod +x script.sh
10      ./script.sh
11      m5 exit
12  fi
13  # otherwise, drop to the terminal
```

x86-npb-benchmarks.py

gem5 > configs > example > gem5_library > x86-npb-benchmarks.py > ...

```
190     # properly.
191
192     command="/home/gem5/NPB3.3-OMP/bin/{}.{}.x;".format(args.benchmark,args.size)\
193         + "sleep 5;" \
194         + "m5 exit;"
195
196     board.set_kernel_disk_workload(
197         # The x86 linux kernel will be automatically downloaded to the
198         # `~/.cache/gem5` directory if not already present.
199         # npb benchamarks was tested with kernel version 4.19.83
200         kernel=Resource(
201             "x86-linux-kernel-4.19.83",
202         ),
203         # The x86-npb image will be automatically downloaded to the
204         # `~/.cache/gem5` directory if not already present.
205         disk_image=Resource(
206             "x86-npb",
207         ),
208         readfile_contents=command,
209     )
```

How to create your kernel binary?

README.md

gem5-resources > src > linux-kernel > README.md > ...

````
```sh
# will create a `linux` directory and download the initial kernel files into it.
git clone https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git

cd linux
# replace version with any of the above listed version numbers
git checkout v[version]

# copy the appropriate Linux kernel configuration file from linux-configs/
cp ../linux-configs/config.[version] .config

make -j`nproc`
```
````

After this process succeeds, the compiled Linux binary, named vmlinux, can be found in the /linux.

If you need the binary in a different ISA than the host, you need to use cross-compilers.

(Link to full resource instructions)

Extending/Modifying gem5 Resources

Extending/Modifying a gem5 resource

Use “**CustomResource**”

x86-npb-benchmarks.py U

materials > using-gem5 > 08-fullsystem > x86-npb-benchmarks.py > ...

```
60  from gem5.components.processors.cpu_types import CPUTypes
61  from gem5.isas import ISA
62  from gem5.coherence_protocol import CoherenceProtocol
63  from gem5.resources.resource import Resource, CustomResource, CustomDiskImageResource
64
```

Extending/Modifying a gem5 resource

Use “**CustomResource**”

x86-npb-benchmarks.py U

materials > using-gem5 > 08-fullsystem > x86-npb-benchmarks.py > ...

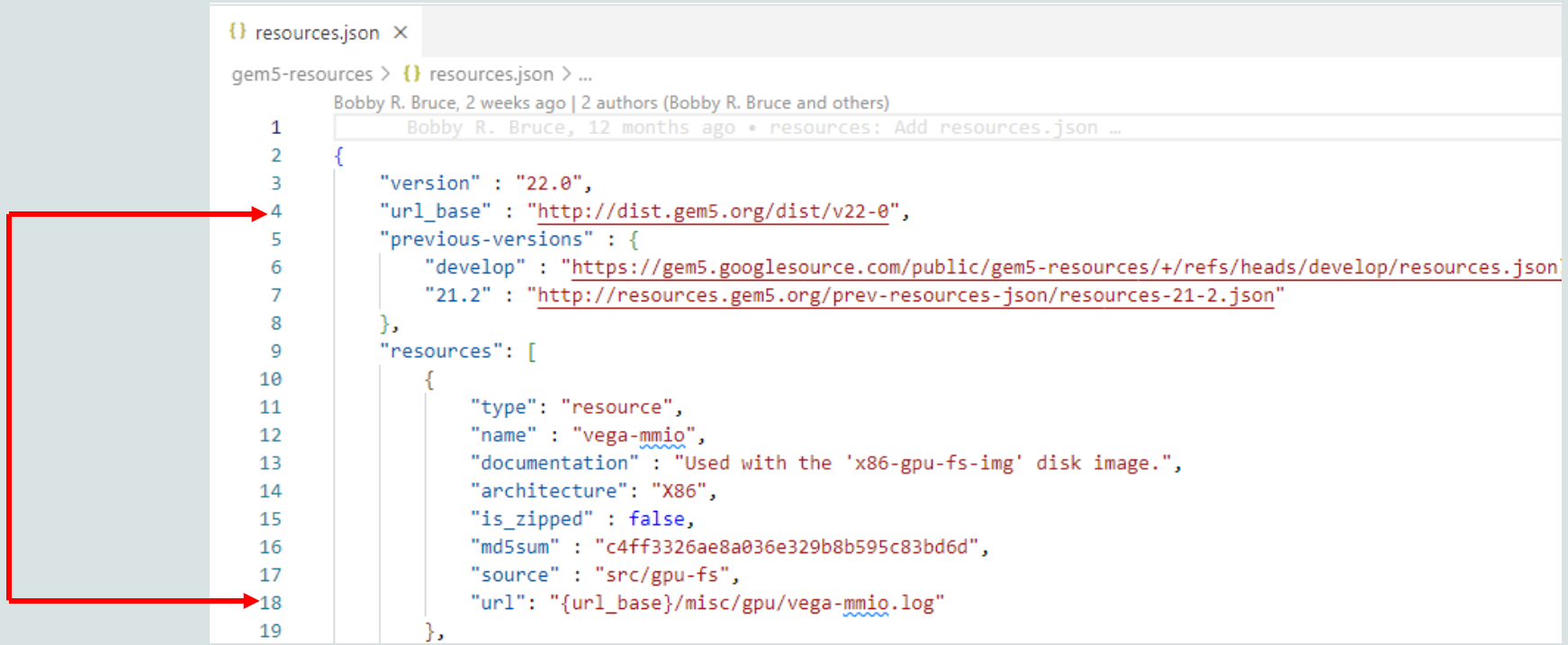
```
196  board.set_kernel_disk_workload(
197      # The x86 linux kernel will be automatically downloaded to the
198      # `~/.cache/gem5` directory if not already present.
199      # npb benchamarks was tested with kernel version 4.19.83
200
201      # kernel=Resource(
202      #     "x86-linux-kernel-4.19.83",
203      # ),
204      kernel=CustomResource(
205          "PATH_TO_YOUR_KERNEL",
206      ),
207
208      # The x86-npb image will be automatically downloaded to the
209      # `~/.cache/gem5` directory if not already present.
210
211      # disk_image=Resource(
212      #     "x86-npb",
213      # ),
214      disk_image=CustomResource(
215          "PATH_TO_YOUR_DISK_IMG",
216      ),
217
218      # disk_image=CustomDiskImageResource(
219      #     "PATH_TO_YOUR_DISK_IMG",
220      # ),
221      readfile_contents=command,
222  )
```

How to get prebuilt resources from gem5 resources?

gem5 resources provides pre-built disk images and kernel binaries for several well-known benchmarks.

In the *resources.json* file, you can find a full list of these pre-built resources.

Full URL
to download



Creating a new gem5 resource

resources.json

gem5-resources > resources.json > ...

Bobby R. Bruce, 2 weeks ago | 2 authors (Bobby R. Bruce and others)

Bobby R. Bruce, 12 months ago • resources: Add resources.json …

```
{
    "version" : "22.0",
    "url_base" : "http://dist.gem5.org/dist/v22-0",
    "previous-versions" : {
        "develop" : "https://gem5.googlesource.com/public/gem5-resources/+/refs/heads/develop/resources.json?format=TEXT",
        "21.2" : "http://resources.gem5.org/prev-resources-json/resources-21-2.json"
    },
    "resources": [
        {
            "type": "resource",
            "name" : "vega-mmio",
            "documentation" : "Used with the 'x86-gpu-fs-img' disk image.",
            "architecture": "X86",
            "is_zipped" : false,
            "md5sum" : "c4ff3326ae8a036e329b8b595c83bd6d",
            "source" : "src/gpu-fs",
            "url": "{url_base}/misc/gpu/vega-mmio.log"
        },
        {
            "type": "resource",
            "name" : "riscv-hello-example-checkpoint",
            "documentation" : "A checkpoint used in 'configs/example/gem5_library/checkpoints/riscv-hello-restore-checkpoint.py'.",
            "architecture": "RISCV",
            "is_zipped" : false,
            "md5sum" : "3a57c1bb1077176c4587b8a3bf4f8ace",
            "source" : null,
            "is_tar_archive" : true,
            "url": "{url_base}/checkpoints/riscv-hello-example-checkpoint.tar"
        },
```

m5term Tool

m5term Tool

The m5term API allows the user to connect to the **simulated console interface** that full-system gem5 provides.

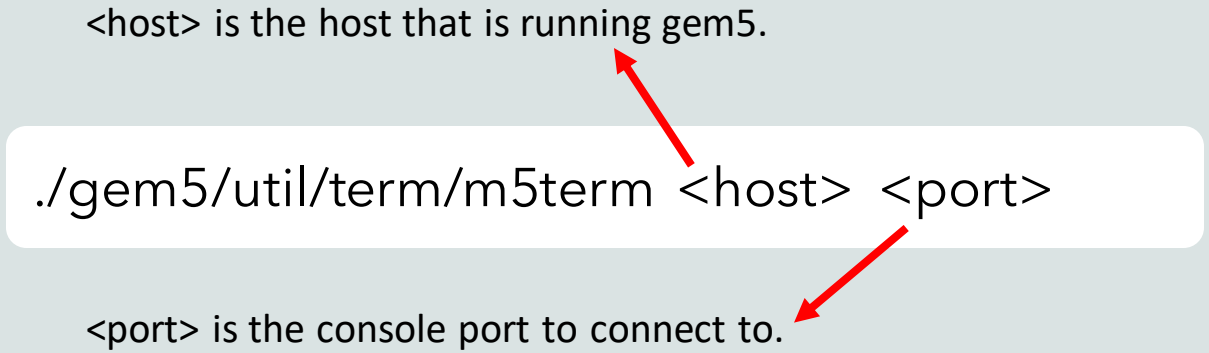
Building m5term

Simply, go to the directory that the API resides, and use “make” command:

```
cd gem5/util/term
make
```

Using m5term

The format to use m5term is as follows:



Look for the following line while gem5's running:

```
"system.platform.terminal: Listening for connections on port <port>"
```

gem5 defaults to using port 3456, if the port is not used.

m5term uses '~' as an escape character. If you enter the escape character followed by a '.', the m5term program will exit.

m5term: Example

riscv-fs.py U

materials > using-gem5 > 08-fullsystem > riscv-fs.py > ...

```
25  # OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
26
27  """
28  This example runs a simple linux boot. It uses the 'riscv-disk-img' resource.
29  It is built with the sources in `src/riscv-fs` in [gem5 resources](
30  https://gem5.googlesource.com/public/gem5-resources).
31
32  Characteristics
33  ---------------
34
35  * Runs exclusively on the RISC-V ISA with the classic caches
36  * Assumes that the kernel is compiled into the bootloader
37  * Automatically generates the DTB file
38  * Will boot but requires a user to login using `m5term` (username: `root`,
39    password: `root`)
40  """
41
42  from gem5.components.boards.riscv_board import RiscvBoard
43  from gem5.components.memory import SingleChannelDDR3_1600
```

Run gem5:

gem5/build/RISCV/gem5.opt materials/using-gem5/08-fullsystem/riscv-fs.py

In the terminal:

```
Beginning simulation!
Global frequency set at 1000000000000 ticks per second
build/RISCV/sim/kernel_workload.cc:46: info: kernel located at: /root/.cache/gem5/riscv-bootloader-
vmlinux-5.10
      0: board.platform.rtc: Real-time clock set to Sun Jan  1 00:00:00 2012
board.platform.terminal: Listening for connections on port 3456
0: board.remote_gdb: listening for remote gdb on port 7000
build/RISCV/arch/riscv/linux/fs_workload.cc:50: info: Loading DTB file: m5out/device.dtb at address
 0x87e00000
build/RISCV/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
```

Run m5term in a separate terminal:

./gem5/util/term/m5term localhost <port>