

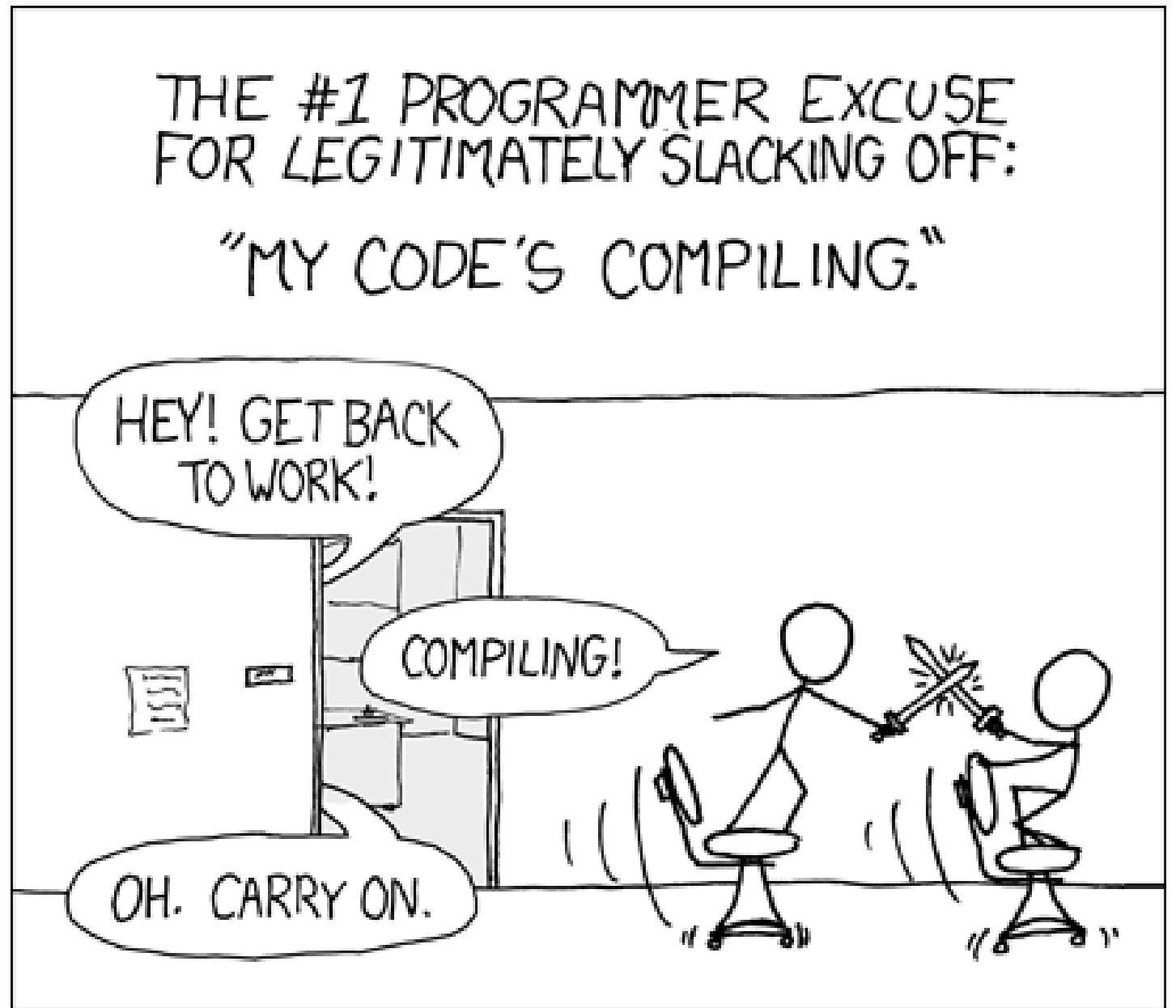
Logistics

- Homework 1 presurvey link will be available tonight. (Dr Lopez Para will distribute the agreements)
- Homework 2 will be assigned Friday and due on Monday Sept 15th by 5:00pm.



"The Russians hacked my homework."

Lecture 4:
Compilers and
Vector
Operations +
Creating User
Accounts



Login to your team server

```
ssh root@129.24.245.{TeamID}
```

Install GNU system compiler

```
yum install g++
```



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

Install GNU system compiler

```
yum install gfortran
```



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

Install GNU system compiler

yum install emacs



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

Install GNU system compiler

man emacs



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

Install GNU system compiler

`man emacs`

You can use whatever text editor you want of course.

Vi is preinstalled on all Linux systems

If you don't already know a linux editor, you can install nano with

`yum install nano`



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

UNIX

UNIPLEXED
INFORMATION
COMPUTING SYSTEM

POSIX (Portable OS Interface) vs GNU

Vi is preinstalled because it was part of the 1970s POSIX Standard that helped define UNIX.

“Do one thing and do it well” philosophy

GNU tools were written to be a free clone of POSIX UNIX

“Everyone should be free to change their software”



Preinstalling emacs would stifle changes



Richard Stallman
GNU (GNU is Not Linux)

GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983

Install GNU Emacs

Copyright 1995, 1999-2021 Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.



GNU software is copyleft – meaning it can be freely distributed and modified, as long as the modified code is also copyleft.

This means software derived from GNU free software must remain free.

That doesn't mean programmers can't make money from derived software but they can't restrict access.

Richard Stallman
GNU (GNU is Not Linux)

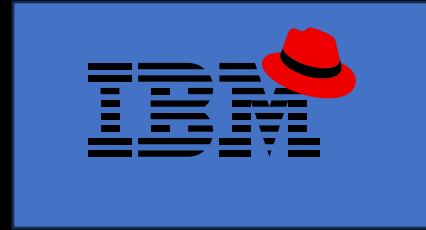
GNU Tools are developed by the Free Software Foundation
Writing free tools based on UNIX utilities since 1983



Install GNU Emacs

Copyright 1995, 1999-2021 Free Software Foundation, Inc.

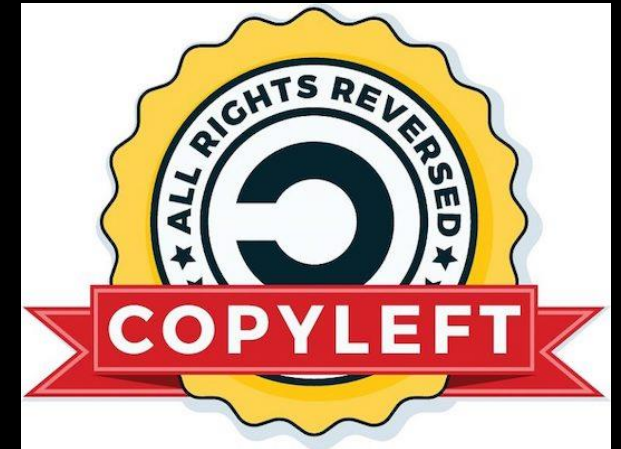
Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.



Richard Stallman
GNU (GNU is Not Linux)



Etc.



IBM owns RedHat Linux. But since it is based on GNU copyleft code they can't keep others from copying it.

Copyleft

```
[root@moonshine ~]# gfortran --version
```

```
GNU Fortran (GCC) 11.4.1 20230605 (Red Hat 11.4.1-2)
```

```
Copyright (C) 2021 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR  
PURPOSE.
```

Install GNU system compiler

```
yum install git
```

```
GIT(1)
```

```
NAME
```

```
git - the stupid content tracker
```

Git Manual

Linus Torvalds

git noun

🔊 /git/

🔊 /git/

(British English, slang)

★ a stupid or unpleasant man

• *What a miserable old git!*

Create User Account

```
[root@moonshine ~]# adduser matthew
```

Create User Account

```
[root@moonshine ~]# adduser matthew
```

```
[root@moonshine ~]# passwd matthew
```

Changing password for user matthew.

New password:

Retype new password:

Create User Account

```
[root@moonshine ~]# adduser matthew
```

```
[root@moonshine ~]# passwd matthew
```

Changing password for user matthew.

New password:

Retype new password:

```
[root@moonshine ~]# exit
```

logout

Login to your team server

```
ssh matthew@129.24.245.{TeamID}
```

Login to your team server

```
[matthew@moonshine ~]$ emacs helloworld.cpp
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    cout << "Hello World" << endl;
```

```
    return 0;
```

```
}
```

Write a simple C++ program

```
-UU-:----F1 helloworld.cpp All L1 (C++//1 Abbrev) -----
```

```
-*- mode: compilation; default-directory: "~/" -*-
```

```
Compilation started at Thu Jan 25 23:15:48
```

```
g++ helloworld.cpp -o helloworld
```

```
Compilation finished at Thu Jan 25 23:15:48
```

```
-UUU:%*--F1 *compilation* All L1 (Compilation:exit [0] [0 0 0]) -----
```

Run our C++ program

```
[matthew@moonshine ~]$ ./helloworld  
Hello World
```

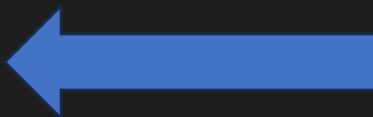
```
program helloworld  
  print *, "Hello World"  
end program helloworld
```

```
-UU-:----F1  helloworld.f90  All L1  (F90) -----
```

```
-* mode: compilation; default-directory: "~/" -*-
```

```
Compilation started at Thu Jan 25 23:57:01
```

```
gfortran helloworld.f90 -o helloworldf
```



```
Compilation finished at Thu Jan 25 23:57:01
```

```
-UUU:%*--F1  *compilation*  All L1  (Compilation:exit [0] [0 0 0]) -----
```

Run our Fortran program

```
[matthew@moonshine ~]$ ./helloworldf  
Hello World
```

Clone SIMD Code

```
git clone https://github.com/gmfricke/SIMD_HW.git
```

Normal non-vector operation array add (one at a time)

```
#define MAX_SIZE (1024 * 1024)

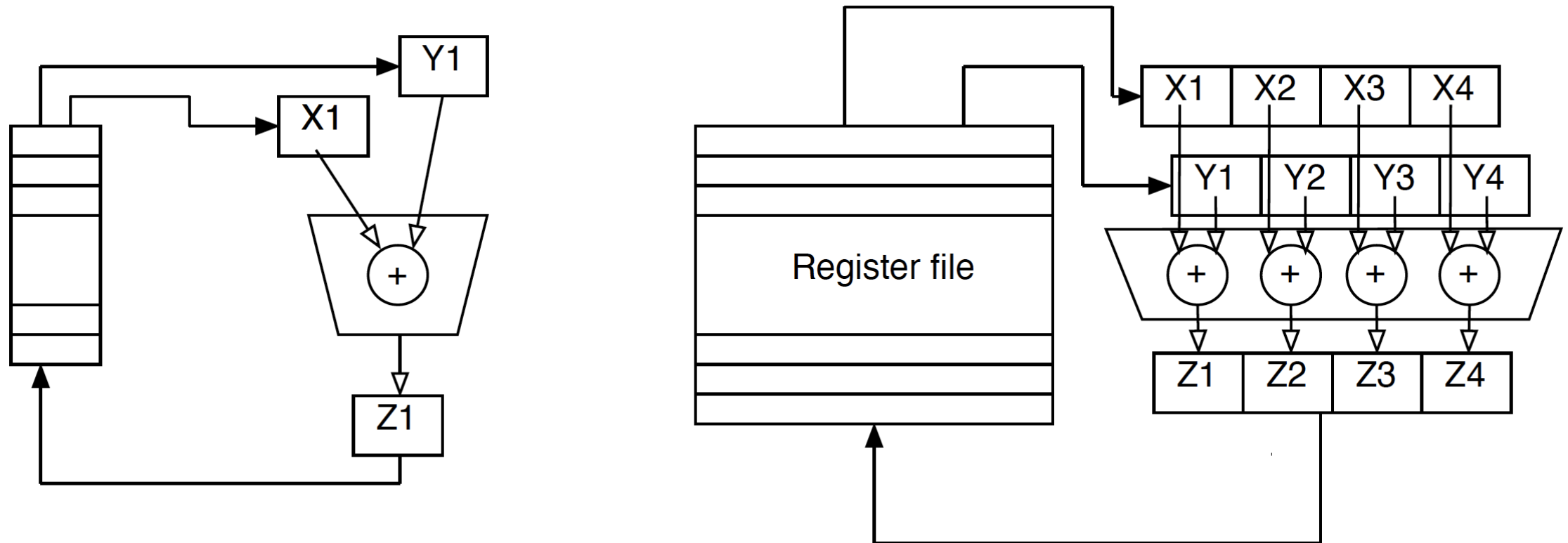
static unsigned short A[MAX_SIZE];
static unsigned short B[MAX_SIZE];

/* Usual vector add – iterate over the arrays one at a time and add */
void add(long size, unsigned short *a, const unsigned short *b) {
    for (int i = 0; i < size; ++i) {
        a[i] += b[i];
    }
}

int main()
{
    /* do it a lot so we can measure */
    for (int i = 0; i < 10e3; i++)
        add(MAX_SIZE, A, B);
    return 0;
}
```

regular_add.c

SSE Vector Operations take 128 bits of data (floating point only), SSE2 adds integer and double operations, AVX 256 bit ops for floating point/doubles but only 128 for ints, AVX2 256 bits (8 ints or floats/4 doubles), AVX512 allows integer and floating point operations. These are examples of Single Operation Multiple Data instructions or SIMD.



GNU C and FORTRAN provide many “intrinsics” so you can access those vector operations. <https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html> provides a complete list.

Vectorized Array-Addition (add 8 unsigned shorts)

```
/* header file for Intel intrinsics */
#include <immintrin.h>
#include <smmmintrin.h>

#define MAX_SIZE (1024 * 1024)

static unsigned short A[MAX_SIZE];
static unsigned short B[MAX_SIZE];

/* vectorized version --- 128-bit registers */
void add_SSE(long size, unsigned short *a, const unsigned short *b) {
    for (int i = 0; i < size; i += 8)
    {
        /* load 128 bits from a */
        /* a_part = {a[i], a[i+1], a[i+2], ..., a[i+7]} */
        __m128i a_part = _mm_loadu_si128((__m128i*) &a[i]);
        /* load 128 bits from b */
        /* b_part = {b[i], b[i+1], b[i+2], ..., b[i+7]} */
        __m128i b_part = _mm_loadu_si128((__m128i*) &b[i]);

        /* Use the SSE 128 bit addition operation */
        a_part = _mm_add_epi16(a_part, b_part);
        _mm_storeu_si128((__m128i*) &a[i], a_part);
    }
}
```

sse_add.c

Vectorized Array-Addition (add 8 unsigned shorts)

We can use these special SSE operations to load 8 values at a time and add them with the `_mm_add_epi16` function.

```
/* load 128 bits from a */
/* a_part = {a[i], a[i+1], a[i+2], ..., a[i+7]} */
__m128i a_part = _mm_loadu_si128((__m128i*) &a[i]);

/* load 128 bits from b */
/* b_part = {b[i], b[i+1], b[i+2], ..., b[i+7]} */
__m128i b_part = _mm_loadu_si128((__m128i*) &b[i]);

/* Use the SSE 128 bit addition operation */
a_part = _mm_add_epi16(a_part, b_part);
_mm_storeu_si128((__m128i*) &a[i], a_part);
```

*You will see a similar “load” execute operation pattern with CUDA programming.

sse_add.c

1. char:

- Size: **1 byte**
- Represents a single character, usually 8 bits (1 byte).

2. short or short int:

- Size: **2 bytes**
- A smaller integer type, usually 16 bits (2 bytes).

3. int:

- Size: **4 bytes** (usually)
- Standard integer type, typically 32 bits (4 bytes) on most systems.

4. long or long int:

- Size: **4 bytes** (on 32-bit systems) or **8 bytes** (on 64-bit systems)
- A longer integer type.

5. long long or long long int:

- Size: **8 bytes**
- An even larger integer type, typically 64 bits.

6. float:

- Size: **4 bytes**
- A single-precision floating point number, typically 32 bits.

7. double:

- Size: **8 bytes**
- A double-precision floating point number, typically 64 bits.

8. long double:

- Size: **8 bytes** or **12-16 bytes** (depending on the system)
- An extended-precision floating point number.

ned shorts)

So with a 128 bit
wide add
instruction
We can load
 $128/16=8$ shorts at
a time but only 4
floats or ints.

Compilation command is a bit complex to make sure the vector operation differences are visible

```
[matthew@moonshine ~]# gcc -g -O1 -std=c99 -mavx2 -mtune=skylake -Wall -fwrapv -fno-strict-aliasing regular_add.c -o regular_add
```

```
[matthew@moonshine ~]# gcc -g -O1 -std=c99 -mavx2 -mtune=skylake -Wall -fwrapv -fno-strict-aliasing sse_add.c -o sse_add
```

Does it matter?

```
[matthew@moonshine ~]# time ./regular_add
```

```
real 0m9.587s  
user 0m9.565s  
sys 0m0.001s
```

```
[matthew@moonshine ~]# time ./sse_add
```

```
real 0m1.447s  
user 0m1.441s  
sys 0m0.002s
```

With SSE the code is
6.6 times faster.

Sudoers

The \$ means you are logged into a user account.
If it were a # you would know it is a root account.

```
[matthew@moonshine ~]$ yum install tar
```

```
Error: This command has to be run with superuser privileges (under the root user on most systems).
```

```
[matthew@moonshine ~]$
```

The user accounts you created can't do administrative tasks.

We will allow the user accounts to run administrative commands. Because:

1. Accountability – everyone logged into root looks the same
2. The root account is often disabled because it is such a juicy target
your root accounts already have thousands of attacks per week.

*more than 4 times due to loop unrolling

Sudoers

```
[root@moonshine ~]# visudo
```

We are going to modify the file that controls account permissions.
/etc/sudoers

Modifying the sudoers file is dangerous. If it is formatted incorrectly you could **lock yourself out of the system entirely**. You can use a regular text editor but the `visudo` command is intended for this purpose.

Visudo does some basic checks to make sure the sudoers file is still valid when you exit.

*more than 4 times due to loop unrolling

Sudoers

```
## Sudoers allows particular users to run various commands as
## the root user, without needing the root password.
##
## Examples are provided at the bottom of the file for collections
## of related commands, which can then be delegated out to particular
## users or groups.
##
## This file must be edited with the 'visudo' command.

## Host Aliases
## Groups of machines. You may prefer to use hostnames (perhaps using
## wildcards for entire domains) or IP addresses instead.
# Host_Alias    FILESERVERS = fs1, fs2
# Host_Alias    MAILSERVERS = smtp, smtp2

## User Aliases
## These aren't often necessary, as you can use regular groups
## (ie, from files, LDAP, NIS, etc) in this file - just use %groupname
## rather than USERALIAS
# User_Alias ADMINS = jsmith, mikem
```

```
## Command Aliases
```

```
"/etc/sudoers.tmp" 120L, 4349B
```

```
1,1
```

```
Top
```

We are going to scroll down to the section where root gets its privileges and copy them.

(a better way would be to add the users to the wheel group but I want you to see the sudoers file)

Sudoers

```
## The COMMANDS section may have other options added to it.
##
## Allow root to run any commands anywhere
root ALL=(ALL)    ALL
matthew ALL=(ALL)    ALL
## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS
```

```
## Allows people in group wheel to run all commands
%wheel ALL=(ALL)    ALL
```

```
## Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL
```

```
## Allows members of the users group to mount and unmount the
## cdrom as root
# %users ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom
```

```
## Allows members of the users group to shutdown this system
# %users localhost=/sbin/shutdown -h now
```

We are going to scroll down to the section where root gets its privileges and copy them.

(a better way would be to add the users to the wheel group but I want you to see the sudoers file)