

## HW01-1 Connect to Linux and Submit Your Solutions

The purpose of this (first) assignment is to explain how to connect to ECN Linux computers and how to submit your solutions.

This program needs 1 file:

1. hw01-1.c

### Connect to ECN Linux Computers

ECE 264 Uses Linux. ECN (Engineering Computer Network) provides IT (Information Technology) services to Purdue's College of Engineering. If you are a student in Purdue Engineering, you should already have an account supported by ECN. If you do not have an account (or this is your first semester), please go to

<https://engineering.purdue.edu/ECN/Support/Accounts/Request>

Please understand that most students do not need to request an account.

After you have an account, you can connect to a Linux computer supported by ECN. You have many options.

In Windows, you can use Command Prompt. Type

```
> ssh youraccount@ececomp.ecn.purdue.edu
```

here > is the prompt and do not type it.

@ececomp.ecn.purdue.edu is a computer accessible if you are off campus.

You need to follow the 2-factor authentication, explained at <https://www.purdue.edu/securepurdue/identity-access/two-factor/index.php> and [https://www.purdue.edu/apps/account/IAMO/BoilerKeyNew/Purdue\\_CareerAccount\\_BoilerKey\\_FAQ.jsp](https://www.purdue.edu/apps/account/IAMO/BoilerKeyNew/Purdue_CareerAccount_BoilerKey_FAQ.jsp)

For example, you may use the **PIN:push** command here

---

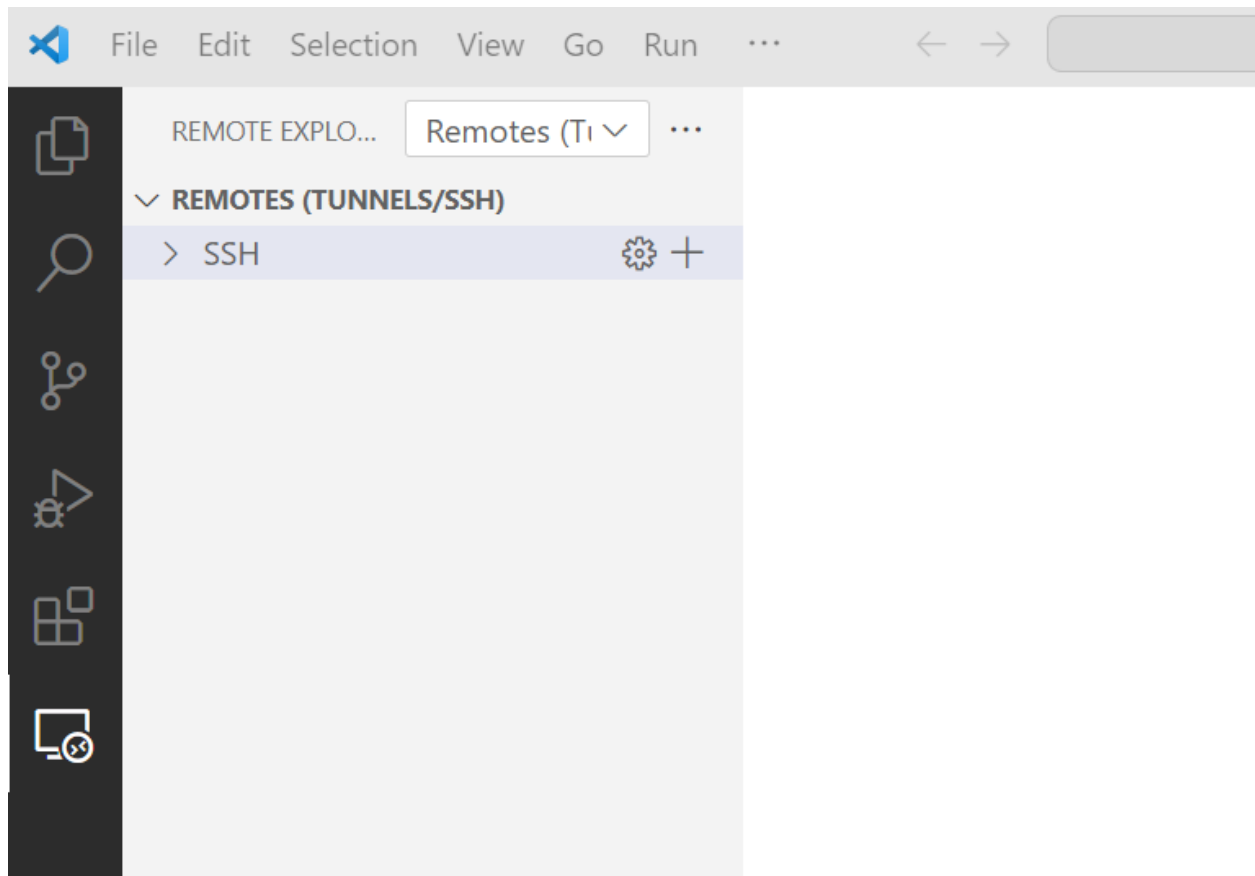
Another option in Windows uses Secure CRT. You can download Secure CRT at <https://web.ics.purdue.edu/~wcrum/cs159/spring10/secureCRT.htm>.

---

The third option is perhaps the most popular: use Visual Studio Code. You can download it at <https://code.visualstudio.com/download>

VSCode supports Windows, Linux, and Mac. Thus, we will spend more time explaining it.

After you install VSCode, start the program. You will see something like



Click the bottom icon on the left side (looks like a computer screen). This is called the "Remote Explorer".

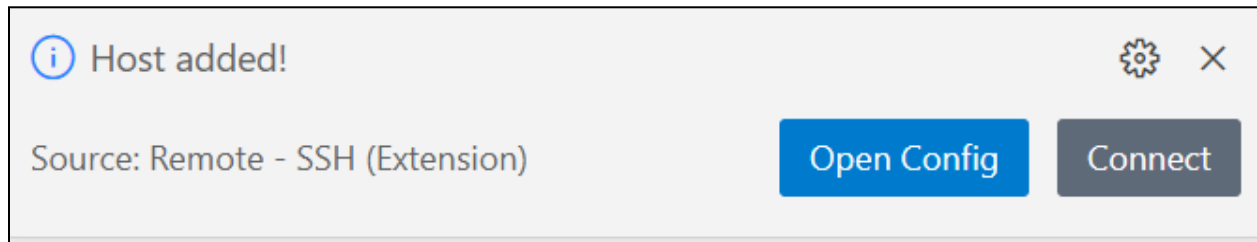
Click + on the right side of SSH. This window appears.

Enter SSH Connection Command
E.g. <code>ssh hello@microsoft.com -A</code>
Press 'Enter' to confirm your input or 'Escape' to cancel

Enter

`ssh youraccount@ececomp.ecn.purdue.edu`

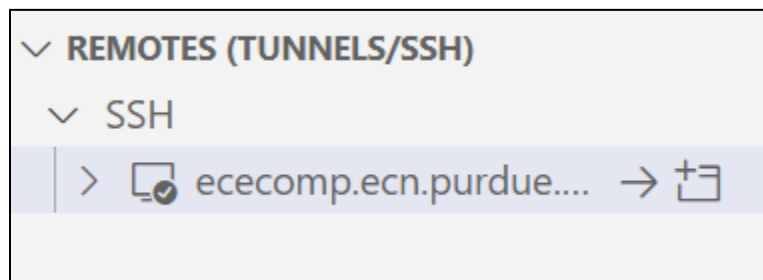
If this is the first time you connect with @ececomp.ecn.purdue.edu, this window will appear



Click Connect

You need to follow the 2-factor authentication.

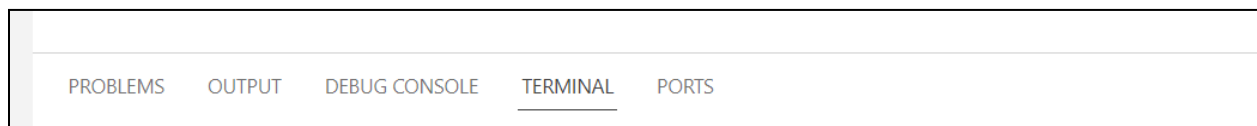
If you see



You have successfully connected to @ ececomp.ecn.purdue.edu

Usually, you want to have a terminal where you can type Linux commands. Click View on the menubar and select Terminal.

You should see



At the terminal, you probably should create a new directory for all programming assignments.

`mkdir ECE264`

Please be aware that Linux is case sensitive. ECE264 is different from ece264. Please do not have space between ECE and 264. The space will create many problems.

## Download Programming Assignment

Type

```
git clone git@github.com:Purdue-ECE264-Spring2024-002/assignments.git
```

This is an example. The actual command (after @github.com) should reflect the correct path. This assignment is mostly reading (and little coding). The purpose of this assignment is to help you understand the tools needed for writing and submitting the programming assignments.

## Programming Assignment 001

This assignment uses one (only one) file for the program. This file is called **hw001.c**. Please read the program carefully and understand it fully.

### Compile hw001.c

hw001.c is written as a text file. Unfortunately, computers do not understand it. This text file needs to be converted into another format, called **executable**, that can be understood by computers. The conversion can be done using the following command typed in a terminal.

```
gcc -std=c99 -g -Wall -Wshadow -pedantic -Wvla -Werror hw001.c -o hw001
```

- -std=c99 means using the C standard announced in 1999
- -g means enabling debugging
- -Wall means enable warning messages. This can identify some common mistakes.
- -Wshadow means detecting shadow variables. Detect shadow variables if your program has any.
- -pedantic means restricting standard C
- -Wvla means detecting variable length array
- -Werror means treating warnings as errors
- hw001.c is the C program (input of gcc)
- -o tells gcc the next word is the name of the output file
- hw001 is the output of gcc. It is an executable.

If you type

```
file hw001
```

You will see something like

```
hw001: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked,  
interpreter /lib64/ld-linux-x86-64.so.2,
```

BuildID[sha1]=cd4bdddffa7f7ee0655b123c1e3f48a0a27078adf, for GNU/Linux 3.2.0, with debug\_info, not stripped

The most important word here is **executable**.

Please always add **-g -Wall -Wshadow** after gcc. They can detect some common mistakes and save you a lot of time. Later, we will explain another tool (called **make**) that can reduce the number of keystrokes.

To run the program, type

```
./hw001 2024 264
```

Adding **./** mean the file hw001 in the current directory (also called folder). It is needed because you may have files of the same in multiple directories.

The terminal will show the sum:

```
2288
```

If you do not provide two numbers or your provide too many numbers, the program has no output.

If you add **> output**, the program's output will be saved in the file called output (you can use any valid name for a file to store the output). This is called redirecting the program's output.

```
./hw001 2024 264 > output
```

Nothing will appear at the terminal. If you type

```
more output
```

You can see the content of the file (2288).

## Submit to Gradescope

A major change in Spring 2024 (Section 2, Instructor: Yung-Hsiang Lu) is the adoption of a new grading system. This system uses Gradescope. Gradescope has several advantages:

- Gradescope is integrated with Brightspace and your scores will appear in Brightspace automatically.
- Gradescope allows students to submit at any time. This greatly reduces the effort of the teaching staff for grading. As a result, the teaching staff can devote more time helping students.

Because of these two advantages, many programming assignments in past semesters are divided into several smaller assignments so that partial credits can be given more easily. In other words, there are more programming assignments but each needs less effort. The total amount of effort of these assignments is unchanged.


Gradescope accepts zip files for submissions; zip is a format of compressed files. Please type

`zip hw001.zip hw001.c`

This creates a zip file from hw001.c (please notice that the destination hw001.zip is the first word after zip).

## Submission Process

Now, you can upload hw001.zip to Gradescope. Log into gradescope and choose this course.


 < ≡  
**ECE 264-Spring-2024-Section002**  
Advanced C Programming  
Dashboard  
Regrade Requests

**ECE 264-Spring-2024-Section002** | Spring 2024  
Course ID: 678185  
**Description**  
files, structures, dynamic structures, recursion. 9-1015AM  
Tuesdays and Thursdays in ARMS 1010  

◆ Name	◆ Status
<u>PA1: Set up and submission</u>	● No Submission

Select "PA1:Set up and submission"

# Submit Programming Assignment

 Upload all files for your submission

## Submission Method

☒  Upload ☐  GitHub ☐  Bitbucket

## Drag & Drop

Any file(s) including .zip. Click to browse.

Cancel

Upload

Upload the zip file.

Add files via Drag & Drop or [Browse Files.](#)

Name	Size	Progress	✕
hw001.c	0.3 KB	<div></div>	✕

Cancel

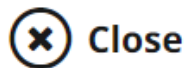
Upload

Gradescope tells you the content of the zip file. The zip file contains one file called hw001.c.

This is the confirmation of successful upload.

# PA1: Set up and submission submitted successfully!

A receipt of your submission has been sent to luyunghsiang@gmail.com. You may be able to resubmit until the due date (**Jan 19 at 5:00PM (EST)**). You will be notified if your grades are made available.



## Grading Submissions

This is the grading result when the submitted program is correct.

Autograder Results

ResultsCode

Check submitted files (10/10)

All required files submitted!

Run gcc (10/10)

gcc succeed

add two numbers (80/80)

```
gcc passed
./hw001 4282 7541 > output
11823
./hw001 2300 152 > output
2452
./hw001 8460 8248 > output
16708
./hw001 2339 4623 > output
6962
./hw001 2861 8072 > output
10933
```

PA1-1: Set up and submission

Student  
Grade Tester

Total Points  
- / 100 pts

Autograder Score  
100.0 / 100.0

Passed Tests  
Check submitted files (10/10)  
Run gcc (10/10)  
add two numbers (80/80)

The correct program should print  $v1 + v2$ . If the program prints  $v1 - v2$  instead, this is the output of grading:



Autograder Results

Results
Code

Check submitted files (10/10)

All required files submitted!

Run gcc (10/10)

gcc succeed

add two numbers (0/80)

```
gcc passed
./hw001 7517 1291 > output
6226

Test Failed: 6226 != 8808
```

PA1-1: Set up and submission

Student  
Grade Tester

Total Points  
- / 100 pts

Autograder Score  
20.0 / 100.0

Failed Tests  
add two numbers (0/80)

Passed Tests  
Check submitted files (10/10)  
Run gcc (10/10)

Gradescope can show the submission history:

Submission History				
#	Submitted On (EST)	Submitters	Score	Active
3	<a href="#">Dec 10 at 8:47 PM</a>	GT	20.0	✓
2	<a href="#">Dec 10 at 4:42 PM</a>	GT	100.0	Activate
1	<a href="#">Dec 9 at 4:06 PM</a>	GT	0.0	Activate

## Settings of Automatic Grading

You do not need to read this section for ECE 264. If you want to understand how programming assignments are graded by Gradescope, please read

[https://help.gradescope.com/article/ujutnle52h-instructor-assignment-programming#creating\\_a\\_programming\\_assignment](https://help.gradescope.com/article/ujutnle52h-instructor-assignment-programming#creating_a_programming_assignment)

How is this assignment graded?

The grading program has the following files:

```
├── requirements.txt
├── run_autograder
├── run_tests.py
├── setup.sh
├── tests
│   ├── test_files.py
│   ├── test_gcc.py
│   └── test_value.py
```

Here are the function and content of each file:

setup.sh: tells the grading system what software tools are needed. For this assignment, the content of setup.sh is

```
#!/usr/bin/env bash
apt-get install gcc
apt-get install -y python3 python3-pip python3-dev
pip3 install -r /autograder/source/requirements.txt
```

This is the content of requirements.txt

```
gradescope-utils>=0.3.1
subprocess32
```

This is the content of run\_autograder

```
#!/usr/bin/env bash

cp /autograder/submission/hw001.c /autograder/source/hw001.c
cd /autograder/source
python3 run_tests.py
```

Please notice that the file name for this assignment must be **hw001.c**. If your program's file name is different, the grading process will not succeed.

This is the content of run\_tests.py

```
import unittest
from gradescope_utils.autograder_utils.json_test_runner import JSONTestRunner
```

```

if __name__ == '__main__':
    suite = unittest.defaultTestLoader.discover('tests')
    with open('/autograder/results/results.json', 'w') as f:
        JSONTestRunner(visibility='visible', stream=f).run(suite)

```

run\_tests.py uses Python's unittest framework. It goes to the tests directory and executes all tests.

For this assignment, there are three tests:

test\_files.py: check whether the correct file **hw001.c** has been submitted

```

import unittest
from gradescope_utils.autograder_utils.decorators import weight
from gradescope_utils.autograder_utils.files import check_submitted_files

class test_files(unittest.TestCase):
    @weight(10)
    def test_files(self):
        """Check submitted files"""
        missing_files = check_submitted_files(['hw001.c'])
        for path in missing_files:
            print('Missing {0}'.format(path))
        self.assertEqual(len(missing_files), 0, 'Missing required file')
        print('All required files submitted!')

```

test\_gcc.py: check whether the program can be compiled successfully. Please notice the flags (-Wall -Wshadow) after gcc.

```

import unittest
from gradescope_utils.autograder_utils.decorators import weight
from gradescope_utils.autograder_utils.files import check_submitted_files
import os

class test_gcc(unittest.TestCase):
    @weight(10)
    def test_gcc(self):
        """Run gcc"""
        compiled = os.system(f"gcc -std=c99 -g -Wall -Wshadow -pedantic -Wvla -Werror hw001.c -o hw001")
        if (compiled != 0):
            print('gcc fail')
        else:
            print('gcc succeed')
        self.assertEqual(compiled, 0)

```

test\_value.py: generate some numbers and check whether the program's output matches the

expected output.

```
import unittest
from gradescope_utils.autograder_utils.decorators import weight
from gradescope_utils.autograder_utils.files import check_submitted_files
import os
import random

class test_values(unittest.TestCase):
    @weight(80)
    def test_values(self):
        """add two numbers"""
        compiled = os.system(f"gcc -g -Wall hw001.c -o hw001")
        if (compiled != 0):
            print('Compile fail')
            self.assertEqual(compiled, 0)
            print('gcc passed')
            # print(os.listdir("."))
            for iter in range(10):
                v1 = random.randrange(10000)
                v2 = random.randrange(10000)
                command = f"./hw001 {v1} {v2} > output"
                print(command)
                runrtv = os.system(command)
                if (runrtv != 0):
                    print("Execution fail")
                    # print(os.listdir("."))
                    self.assertEqual(runrtv, 0)
                    f = open('output', 'r')
                    value = int(f.read())
                    print(value)
                    self.assertEqual(value, v1 + v2)
```