TDDE18 & 726G77

Expressions

Lab soft deadlines

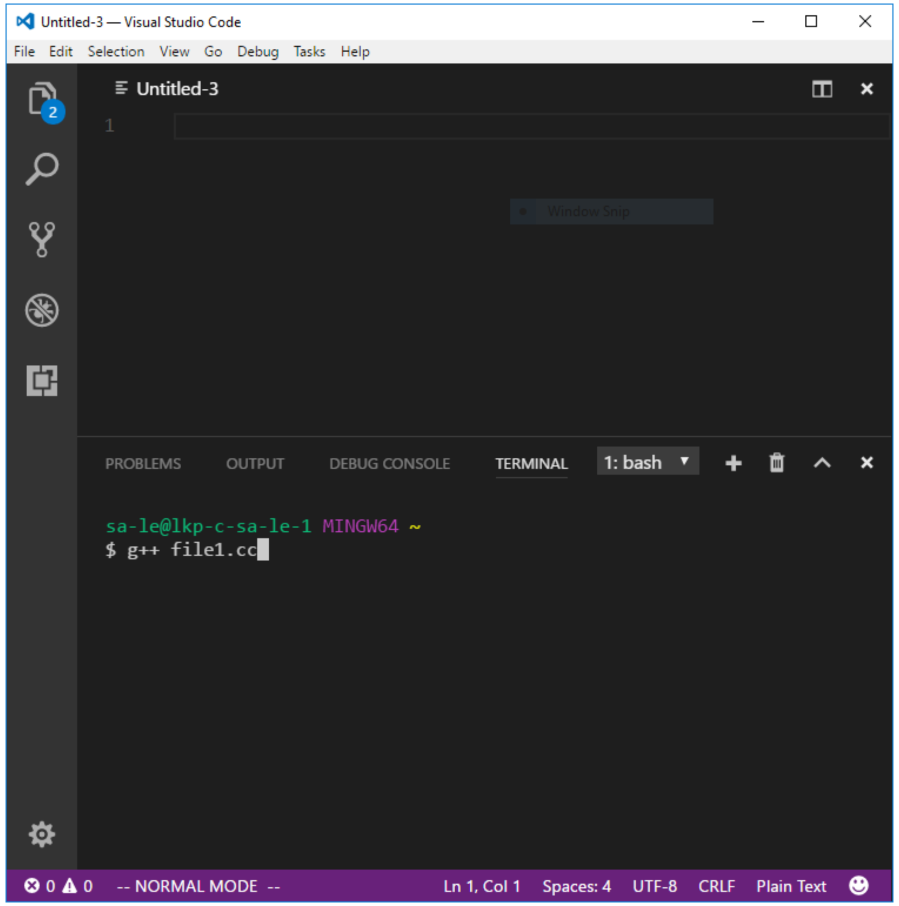
- Bonus time to the exam for higher grade
- 5 extra minute per deadline
- 1 deadline per lab (1 – 7)
- 1 complementary work per lab
- You must demonstrate your work for the assistant.
  - They will give you the one time password needed to submit your work

Lesson 1

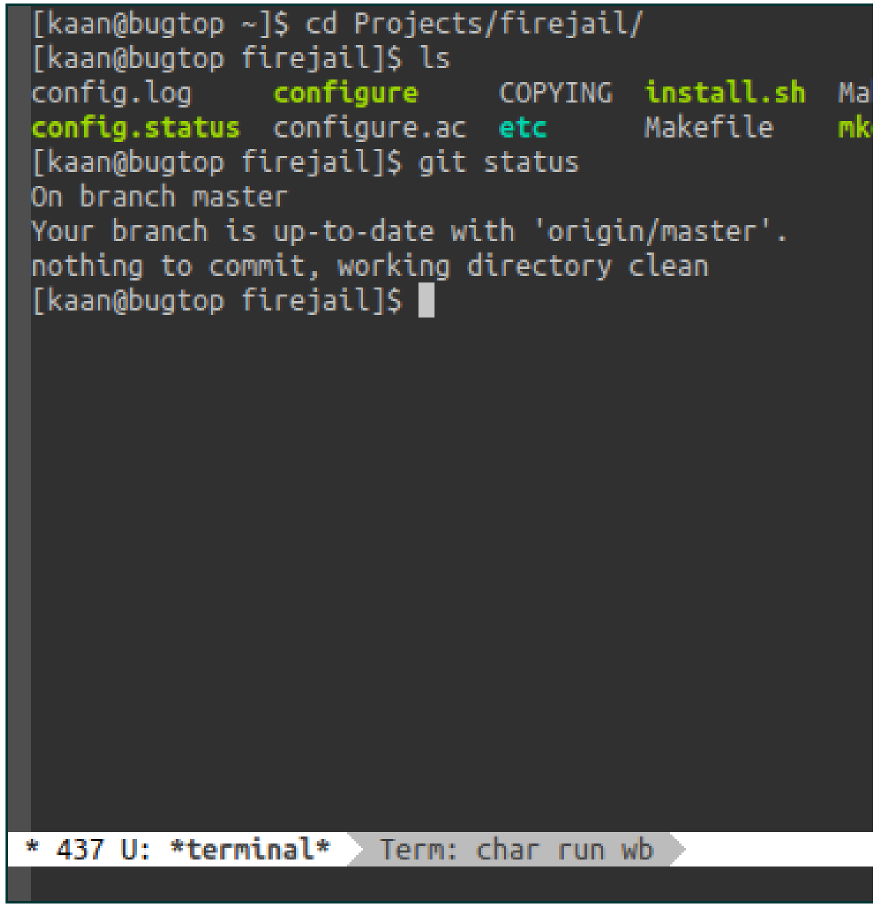
- Interactive where you will solve programming problems with your assistant
- 3 rooms in Swedish / 1 room in English – check the schedule
- First lesson will be kind of basic and if you feel that todays lecture is too easy then you might not need to come.

Tool tip of the lecture

- Terminal inside vscode & emacs


Untitled-3 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help
Untitled-3
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
0 0 -- NORMAL MODE -- Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text

```
sa-le@lkp-c-sa-le-1 MINGW64 ~
$ g++ file1.cc
```



```
[kaan@bugtop ~]$ cd Projects/firejail/
[kaan@bugtop firejail]$ ls
config.log     configure     COPYING  install.sh  Ma
config.status  configure.ac  etc      Makefile    mk
[kaan@bugtop firejail]$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
[kaan@bugtop firejail]$
```

Todays lecture

```
double sum{0};
int integer{0};
for (int i{0}; i < 5 and cin >> integer; ++i) {
     if (integer % 2) {
          sum += 100 / static_cast<int>(integer);
     }
}
cout << “The sum is: “ << sum << endl;

User enter: 1 2 3 4 5
```

Conditional statements: if/else if/else

```
if (some logical statement) {
        do this
}
else if (some other logical statement) {
        do this instead
}
else {
        when all else fails do this
}
```

Comparison and Logical operators

- `a == b`
- `a != b`
- `a < b`
- `a <= b`
- `a > b`
- `a >= b`

```
a = 1, b = 2
```

- `a == b and c != b`
- `a == b or a == c`
- `!a`
- `&& is equivalent to and`
- `|| is equivalent to or`

```
c = 3, d = 4
```

Code example

```
int a{2};
int b{2};
if (a < b) {
     cout << “This will not be executed” << endl;
}
```

Code example

```
int a{2};
int b{2};
if (a > b and a == b) {
     cout << “This will not be executed” << endl;
}
```

Code example

```
int a{2};
int b{2};
if (a != b or a) {
     cout << “This is true” << endl;
}
```

loops

- for loops
- while loops
- do-while loops

- Which one to use depends on purpose and readability

For loops

- You know exactly how many times you want to loop

```
for (initializing; conditional statement; incrementing) {
      body
}
```

Code example

```
for (int i{0}; i < 5; ++i) {
     cout << i << “ “;
}
```

While loops

- When you do not know how many times it will run

```
while (conditional statement) {
     body
}
```

Code example

```
string str{};
while (cin >> str) {
      cout << str << endl;
}
```

cin >> str returns false when there is nothing in the input buffer

Do-While loop

- Run the body at least once

```
do {
        body
} while (conditional statement);
```

Code example

```
do {
    cout << “Enter a number between 0 and 10: “;
    cin >> integer;
} while (integer < 0 and integer > 10);
```

Arithmetic operators

`+, -, *, /, %`

Example:

- `1 + 3`
- `a – b`
- `c * d`
- `10.0 / 3`
- `3 % 2`

Arithmetic operators

```
+=, -=, /=, *=
++, --

Example:
a += 4; => a = a + 4
b++ => b = b + 1;
--a => a = a – 1;
```

```
int a{0};
int b{1};

int c{a++}; // What is c?
c = --b; // What is c?
```

Type casting

Problem:

3 / 2 = 1 // integer division

but

3 / 2.0 = 1.5

Example:

```
int a{3};
int b{2};

cout << a / b << endl;
```

```
output: 1
```

Type casting

static_cast<*new type*>(*input*) *// will return a value of the new type*

Example:

- `static_cast<int>(‘a’); // 65 due to ascii table`
- `static_cast<double>(1); // float value 1.0`

- dont use c-cast eg: (double)a

Commenting

```
// line comment



/*
   multiline
   comment
*/
```