

DASL-100.2 C++ Programming and Linux



Week 4-1

 IDE (Integrated Development Environment)
 Vector
 Thread

Instructor: Truc Tran

02-18-2023

DASL-100.2 C++ Programming and Linux



1. IDE (Integrated Development Environment)

- IDE is a software application that integrates a code editor, compiler, debugger, and other tools to facilitate the development, testing, and debugging of C++ programs. It offers many features include:
 - Editing source code

HOWARD R. HUGHES

- Building executables, and debugging
- Syntax highlighting, auto complete

VEERING

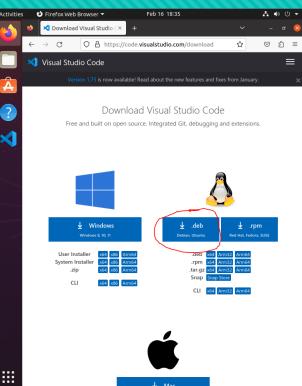
• Common IDE are Visual Studio, Esclipse, Visual Studio Code, Code::Blocks or Clion.



Instrume DASL-100.2 College of C++ Programming and Linux 1. IDE (Integrated Development Environment)



- We will be using Visual Studio Code:
 - Go to: <u>https://code.visualstudio.com/download</u>
 - Click on .deb button (This is for linux ubuntu)





DASL-100.2 C++ Programming and Linux 1. IDE (Integrated Development Environment)



Installing Visual Studio Code in Ubuntu:

JEERING

HOWARD R. HUGHES

- Check for the downloaded file in Downloads folder (code_1.75.1-1675893397_amd64.deb)
- Open terminal and type in the command: sudo dpkg -i code_1.75.1-1675893397_amd64.deb 0

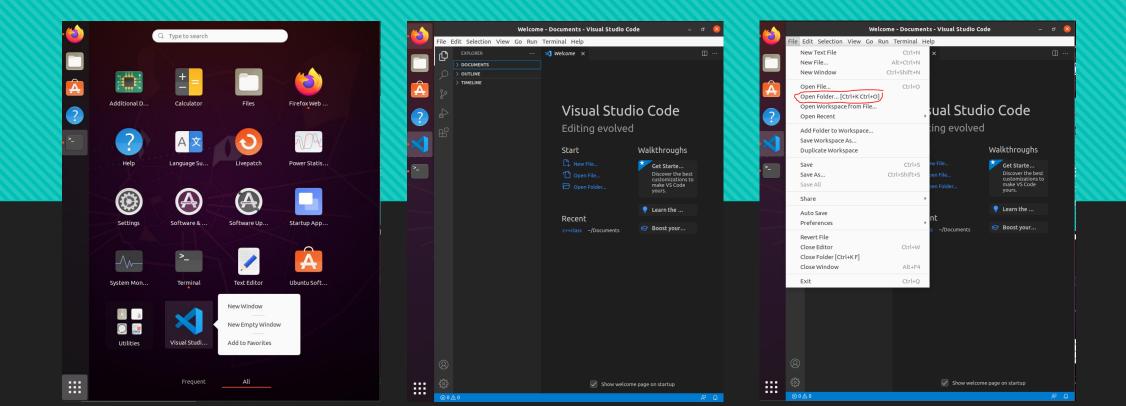
. 台	〈 〉 û Home	Downloads 🕶 🔍 🛛 🔍 🗮 💌 🗮 🗕 🗆 🙁	6	I+1 ubuntu20045@ubuntu: ~/Downloads Q ≡ _ □ ×
	৩ Recent ★ Starred ঐ Home	code_ 1.75.1- 167589339		ubuntu20045@ubuntu:~\$ lsbooleanfunctionclasswithprotectedDownloadssnapbooleanfunction.cppclasswithprotected.cppintfunctionstructcharfunctionclasswithpublicintfunction.cppstruct.cppcharfunction.cppclasswithpublic.cppMusicTemplates
A	Desktop	7_amd64. deb		classwithprivate Desktop Pictures Videos classwithprivate.cpp Documents Public
?	Documents Downloads			ubuntu20045@ubuntu:~\$ cd Downloads ubuntu20045@ubuntu:~/Downloads\$ sudo dpkg -i code_1.75.1-1675893397_amd64.deb [sudo] password for ubuntu20045:
⋈	Music Pictures			(Reading database 163647 files and directories currently installed.) Preparing to unpack code_1.75.1-1675893397_amd64.deb Unpacking code (1.75.1-1675893397) over (1.75.1-1675893397)
• >_	Uideos			Setting up code (1.75.1-1675893397) Processing triggers for gnome-menus (3.36.0-1ubuntu1) Processing triggers for desktop-file-utils (0.24-1ubuntu3)
	 Trash + Other Locations 		>_	Processing triggers for mime-support (3.64ubuntu1) Processing triggers for shared-mime-info (1.15-1)
				ubuntu20045@ubuntu:~/Downloads\$
		"code 1 75 1-1675893397 amd64 deb" selected (98 2 MB)		

DASL-100.2 VEERING C++ Programming and Linux



1. IDE (Integrated Development Environment)

- Open applications and add Visual Studo Code to Favorite.
- Run Visual Studio Code > Click File > Click Open Folder



DASL-100.2 C++ Programming and Linux



1. IDE (Integrated Development Environment)

- Navigate to Documents folder > Click add folder (top left icon) > Create a "c++ class" folder.
- Now we can create new .cpp files by clicking "New Files..".
- The terminal can also be accessed in Visual Studio Code.

JEERING

		Open Folder	Q Open		ocuments - Visual Studio Code 🛛 😑 🙁	<u> </u>	vector.cpp - Documents - Visual Studio Code	- 7 😣			op - Documents - Visual Studio C	iode – ơ (8
	③ Recent	→ lo home 🏠 ubuntu20045 Documents		File Edit Selection View Go Rur	Terminal Help		File Edit Selection View Go Run Terminal Help			File Edit Selection View Go Run			
							EXPLORER ···· C· vector.cpp ×	ш…		EXPLORER	New Terminal	Ctrl+Shift+`	
	습 Home	Name v Size	Type Modified Yesterday				✓ DOCUMENTS			V DOCUMENTS	Split Terminal	Ctrl+Shift+5	
	Desktop	Criclass .	resteredy	New File.			G. vector.cop			G vector.cpp	Run Task	et luel 10 un	
A	Documents			🛃 🖉		A			A		Run Build Task Run Active File	Ctrl+Shift+B	
	Downloads										Run Selected Text		
				? ⁴		?			?		Show Running Tasks		
	Pictures										Restart Running Task		
									\mathbf{X}		Terminate Task		
	Videos										Configure Tasks		
• ~	+ Other Locations			. 2-		>_			• ^{>} -		Configure Default Build Task.		
													4
													4,
												∑ bash +	×
											○ ubuntu20045@ubuntu:~/Docu	nents\$	
					O Got a moment to help the VS Code team? Please tell us about ×								
					your experience with VS Code so far.								
				> OUTLINE	Give Feedback Remind Me later		Construction outline		:::	ج boutline کر کر ک			Γ.
					R C		-U- > TIMELINE	UTF-8 LF C++ 🔊 🕼		> TIMELINE © 0 \triangle 0	Ln 1, Col 1	Spaces: 4 UTF-8 LF C++ 🔗 I	<u>c</u>

UNIV HOWARD R. HUGHES College of ENGINEERING

DASL-100.2 C++ Programming and Linux



2. Vector

- Recall array is a collection of elements of the same data type stored in contiguous memory locations.
- In C++, vector is a <u>dynamically resizable</u> container that can store a collection of elements of a specific data type. Similar to std::string, std::vector is also a class that is part of the Standard Template Library (STL) and is defined in the <vector> header file. The common member functions of the vector class include:
 - push_back(): adds an element to the end of the vector.
 - pop_back(): removes the last element of the vector.
 - size(): returns the number of elements in the vector.
 - empty(): checks if the vector is empty or not. It returns true if the vector is empty and false otherwise.
 - clear(): removes all elements from the vector.
 - insert(): inserts an element at a specified position in the vector.
 - erase(): removes an element at a specified position in the vector.
 - front(): returns the first element of the vector.
 - back(): returns the last element of the vector.
 - begin(): returns an iterator to the beginning of the vector.
 - end(): return an integrator to the end of the vector.

DASL-100.2 Howard R. HUGHES College of ENGINEERING C++ Programming and Linux



2. Vector

Feature	Vector	Array
Size	Dynamic sizing	Size is fixed
Inserting elements	Efficient, with functions like push_back and insert	Not efficient, elements need to be shifted to make room for new ones
Removing elements	Efficient, with functions like pop_back and erase	Not efficient, elements need to be shifted to close the gap
Memory management	Managed automatically, memory is allocated and deallocated as needed	Managed manually, requires explicit allocation and deallocation of memory
Standard Library support	More built-in functions, more functionality	Fewer built-in functions, limited functionality

DASL-100.2 College of C++ Programming and Linux



Drones and Autonomous Systems Lab @ UNLV

2. Vector

UN

HOWARD R. HUGHES

-			vector.cpp - Documents - Visual Studio Co				17 18	<pre>std::cout << "Vector's element 2 value: " << vec[2] << std::endl;</pre>
	File	Edit Selection View						<pre>std::cout << "Vector's first element value: " << vec.front() << std::endl; std::cout << "Vector's first last value: " << vec.back() << std::endl;</pre>
	l (C)	EXPLORER	© vector.cpp ×				21 22	<pre>vec.insert(vec.begin(), 0);</pre>
		✓ DOCUMENTS ✓ c++class	c++class>				23	
		≣ vector	2 #include <vector></vector>					<pre>std::cout << "Vector's size:" << vec.size() << std::endl;</pre>
	း ရှိ	C vector.cpp	<pre>4 int main(){ 5 std::vector<int> vec;</int></pre>				PROBLEMS	S OUTPUT DEBUG CONSOLE TERMINAL
?	\$		6 7 std::cout << "Vector's size:" << vec.size() << std::endl;				• ubuntu2	20045@ubuntu:~/Documents/c++class\$ g++ vector.cpp -o vector 20045@ubuntu:~/Documents/c++class\$ ls
	6		8 9 vec.push_back(10);				• ubuntu2	vector.cpp 20045@ubuntu:~/Documents/c++class\$./vector
			10 vec.push_back(50); 11 vec.push_back(100);				Vector'	's size:0 's size:3 's element 0 value: 10
			12 13 std::cout << "Vector's size:" << vec.size() << std::endl; 14				Vector' Vector'	's element 1 value: 50 's element 2 value: 100
			<pre>15 std::cout << "Vector's element 0 value: " << vec[0] << std::endl;</pre>					's first element value: 10 's first last value: 100
			<pre>16 std::cout << "Vector's element 1 value: " << vec[1] << std::endl; 17 std::cout << "Vector's element 2 value: " << vec[2] << std::endl;</pre>					's size:4 's element 0 value: 0
			<pre>18 19 std::cout << "Vector's first element value: " << vec.front() << std::endl;</pre>				Vector'	's element 1 value: 10 's element 2 value: 50
			<pre>20 std::cout << "Vector's first last value: " << vec.back() << std::endl;</pre>		8		Vector'	's element 3 value: 100
			21 22 vec.insert(vec.begin(), θ);		Solution > OUTLINE		o ubuntu2	20045@ubuntu:~/Documents/c++class\$
			23	:::	STIMELINE SOA 0	E		
			24 std::cout << "Vector's size:" << vec.size() << std::endl; 25	•	Ø Ø Æ Ø			
			<pre>26 std::cout << "Vector's element 0 value: " << vec[0] << std::endl; 27 std::cout << "Vector's element 1 value: " << vec[1] << std::endl;</pre>					
			<pre>28 std::cout << "Vector's element 2 value: " << vec[2] << std::endl;</pre>					
			<pre>29 std::cout << "Vector's element 3 value: " << vec[3] << std::endl; 30 return 0;</pre>					
			31 }					

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

ubuntu20045@ubuntu:~/Documents/c++class\$

> OUTLINE

> TIMELINE

HOWARD R. HUGHES College of ENGINEERING C++ Programming and Linux



3. Thread

- In C++, code is executed in a top-to-bottom flow, meaning that the statements in a program are executed in the order in which they appear. The code is compiled into machine code by the compiler and executed by the operating system, which reads the code line by line and performs the operations specified by each statement.
- In this example, "Read_Sensor_Data" function needs to wait for the "Read_Servo_Position" function to finish first before it can start. This can cause a delay in your robotic system. You wish your control system to operate in a continuous time frame and all the components are working "Independently".
- In C++, a thread is a lightweight execution context that can run in parallel with other threads. A thread can execute code independently from the main program, and multiple threads can run concurrently to perform tasks simultaneously.
- Threads are commonly used to improve the performance and responsiveness of programs by parallelizing workloads across multiple CPUs or processor cores.

	#Include /Instream/
	#include <unistd.h></unistd.h>
3	
	<pre>void Read_Servo_Position(){</pre>
5	int value;
6	for (int i = 0; i < 10; i++){
7	usleep(100000);
8	value = i;
9	<pre>std::cout << "Servo position:" << value << std::endl;</pre>
10	
 11	}
12	
	<pre>void Read_Sensor_Data(){</pre>
	int data;
	for (int i = 0; i < 10; i++){
	usleep(100000);
17	data = i;
	<pre>std::cout << "Sensor data:" << data << std::endl;</pre>
	}
	int main(){
	Read_Servo_Position();
	Read_Sensor_Data();
	return 0;

UNIV HOWARD R. HUGHES College of ENGINEERING

DASL-100.2 C++ Programming and Linux



3. Thread

- The std::thread class is not specifically part of the STL, it is part of the C++ Standard Library, which includes a variety of libraries for different purposes. The common member functions of the thread class include:
 - thread(): the default constructor, which creates a thread object that is not associated with any thread of execution.
 - join(): a member function that blocks the calling thread until the associated thread has completed execution.
 - detach(): a member function that allows the associated thread to execute independently of the thread that created it.
 - get_id(): a member function that returns the unique identifier of the thread.
 - joinable(): a member function that returns true if the thread object is associated with a thread of execution, and false otherwise.

DASL-100.2 College of ENGINEERING C++ Programming and Linux



3. Thread

HOWARD R. HUGHES

• Let's demonstrate the code without using thread.

File Edit Selection View Go Run Terminal Help	·
EXPLORER ··· & vector.cpp & thread.cpp = vector & nothread.cpp × PROBLEMS C	
<pre>vocuments v=c+cdass C=nothread.cpp v=c+cdass C=nothread.cpp v=c+cdass d= #include <cistream> v=c+cdass d= #include <cistream> v=include <unistd.h> v=ubuntu20045 v=c+cdass d= void Read_Servo_Position(){ v=c+cdass d= void Read_Servo_Position(); Red Servo_Position(); Red Servo_P</unistd.h></cistream></cistream></pre>	tion:1 tion:2 tion:3 tion:4 tion:5 tion:6 tion:7 tion:8 tion:9 a:0 a:1 a:2 a:3 a:4 a:5 a:6 a:7 a:8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

bubuntu20045@ubuntu:~/Documents/c++class\$

HOWARD R. HUGHES College of ENGINEERING

DASL-100.2 C++ Programming and Linux



3. Thread

• Now using thread. Note that we need to link the "pthread" library using "-pthread" at the end of g++ complier.

File	Edit Selection View Go Run	Terminal Help					
Ch	EXPLORER ····	C vector.cpp C thread.cpp × ≡ vector C nothread.cpp	PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL				
File C S H	EXPLORER DOCUMENTS C++class C nothread.cpp E thread C thread.cpp E vector C vector.cpp	G: vector.cpp G: thread.cpp E vector G: nothread.cpp 1 #include <iostream> 2 #include <unistd.h> 3 #include <thread> 4 5 void Read_Servo_Position(){ 6 int value; 7 for (int i = 0; i < 10; i++){ 8 usleep(100000); 9 value = i; 10 std::cout << "Servo position:" << value << std::endl; 11 } 12 } 13 14 void Read_Sensor_Data(){ 15 int data; 16 for (int i = 0; i < 10; i++){ 17 usleep(100000); 18 data = i; 19 std::cout << "Sensor data:" << data << std::endl; 20 } 21 } 22 int main()[] 23 int main()[] 24 std::thread thread1(Read_Sensor_Data); 25 std::thread thread2(Read_Sensor_Data); 26 thread1.join(); 29 thread2.join();</thread></unistd.h></iostream>	<pre>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Ubuntu20045@ubuntu:~/Documents/c++class\$ g++ thread.cpp -o thread -pthread ubuntu20045@ubuntu:~/Documents/c++class\$ ls nothread.cpp thread thread.cpp vector vector.cpp ubuntu20045@ubuntu:~/Documents/c++class\$./thread Servo position:0 Sensor data:0 Servo position:1 Sensor data:2 Servo position:2 Servo position:2 Servo position:4 Servo position:5 Sensor data:6 Servo position:6 Sensor data:7 Servo position:7 Sensor data:8 Servo position:8 Sensor data:9 Servo position:9 ubuntu20045@ubuntu:~/Documents/c++class\$</pre>				
		30 return 0; 31 }					