



# **SRI RAMAKRISHNA ENGINEERING COLLEGE**

[Educational Service: SNR Sons Charitable Trust]

[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]

[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]

[ISO 9001-2015 Certified and all eligible programmes Accredited by NBA]

VATTAMALAIPALAYAM, N.G.G.O. COLONY POST,

COIMBATORE – 641 022



## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **20EC276– EMBEDDED SYSTEMS AND INTERNET OF THINGS LABORATORY**

### **LAB RECORD**

**ACADEMIC YEAR: 2022-2023**

**BATCH: 2020-2024**

**APRIL 2023**



# **SRI RAMAKRISHNA ENGINEERING COLLEGE**

[Educational Service: SNR Sons Charitable Trust]

[Autonomous Institution, Reaccredited by NAAC with 'A+' Grade]

[Approved by AICTE and Permanently Affiliated to Anna University, Chennai]

[ISO 9001:2015 Certified and all eligible programmes Accredited by NBA]

VATTAMALAIPALAYAM, N.G.G.O. COLONY POST, COIMBATORE – 641 022.



## **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### **BONAFIDE CERTIFICATE**

**Certified that this is the bonafide record of works done by Mr./Ms.**  
**\_\_\_\_\_ in 20EC276– EMBEDDED SYSTEMS AND**  
**INTERNET OF THINGS LABORATORY of this Institution for VI Semester during**  
**the Academic Year 2022 – 2023.**

#### **Faculty In-Charge**

Dr. B. MATHIVANAN Asso.Prof/CSE

#### **HOD – CSE**

Dr. A. GRACE SELVARANI, Prof/Head

**Date:**

**Register Number:** \_\_\_\_\_

**Submitted for the VI Semester B.E.-CSE Practical Examination held on \_\_\_\_\_**  
**during the Academic Year 2022 – 2023.**

**Internal Examiner**

**Subject Expert**

## INDEX

Exp. No	Date	Title of the Experiment	Page No	Faculty signature
1.	13/12/2022	STUDY OF VARIOUS EMBEDDED CODING STANDARDS		
2.	20/12/2022	DEMONSTRATION OF MISRA C STANDARD		
3.	27/12/2022	DEMONSTRATION OF CERT C STANDARD		
4 A.	03/01/2023	LINUX INSTALLATION PROCEDURE		
4 B.	03/01/2023	LINUX KERNEL COMPILATION		
5.	10/01/2023	DEMONSTRATION OF LINUX COMMANDS		
6.	24/01/2023	UTILIZATION OF GNU TOOL CHAINS OR EFFECTIVE SYSTEM PROGRAMMING		
7.	31/01/2023	BROWSING SOURCE PROGRAMS USING CSCOPE TOOL		
8.	04/02/2023	CONSTRUCT CHARACTER ORIENTED DEVICE DRIVERS		
9 A.	07/02/2023	IMPLEMENTATION OF TASK MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II		
9 B.	14/02/2023	IMPLEMENTATION OF INTERRUPT MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II		
10 A.	21/02/2023	DEVELOPMENT OF BLUETOOTH INTERFACING USING MSP430 LAUNCHPAD		
10 B.	28/02/2023	DEVELOPMENT OF ESP8266 INTERFACING (WIFI) USING MSP430 LAUNCHPAD		
11.	14/03/2023	MULTIPLE LED BLINKING USING TI CC3200 LAUNCHPAD		
12.	21/03/2023	DESIGN OF IOT APPLICATIONS WITH SENSORS USING TI CC3200 LAUNCHPAD		

EX. NO: 1

DATE:13/12/22

## STUDY EXPERIMENT ON EMBEDDED C CODING STANDARD

### AIM:

To study about the various Embedded C coding standard.

### INTRODUCTION:

Barr Group's Embedded C Coding Standard was designed specifically to reduce the number of programming defects in embedded software. By following this coding standard, firmware developers not only reduce hazards to users and time spent in the debugging stage of their projects but also improve the maintainability and portability of their software. Together these outcomes can greatly lower the cost of developing high-reliability embedded software.

This "BARR-C" coding standard is different from other coding standards. Rather than being based on the stylistic preferences of the authors, the rules in this standard were selected for their ability to minimize defects.

### DIFFERENCE BETWEEN C AND EMBEDDED C:

PARAMETERS	C	EMBEDDED C
GENERAL	<ul style="list-style-type: none"><li>C is a general purpose programming language, which can be used to design any type of desktop based applications. It is a type of high level language.</li><li>.</li></ul>	<ul style="list-style-type: none"><li>Embedded C is simply an extension C language and it is used to develop microcontroller-based applications. It is nothing but an extension of C.</li><li>.</li></ul>
DEPENDENCY	<ul style="list-style-type: none"><li>C language is hardware independent language.</li><li>C compilers are OS dependent.</li><li>.</li></ul>	<ul style="list-style-type: none"><li>Embedded C is fully hardware dependent language. Embedded C is OS</li><li>independent.</li></ul>
COMPILER	<ul style="list-style-type: none"><li>For C language, the standard compilers can be used to compile and execute the program.</li><li>Popular Compiler to execute a C language program are:<ul style="list-style-type: none"><li>o GCC (GNU Compiler collection)</li><li>o Borland turbo C,</li></ul></li></ul>	<ul style="list-style-type: none"><li>For Embedded C, a specific compilers that are able to generate particular hardware/micro-controller based output is used.</li></ul>

	<ul style="list-style-type: none"> <li>○ Intel C++</li> </ul>	<ul style="list-style-type: none"> <li>• Popular Compiler to execute a Embedded C language program are: <ul style="list-style-type: none"> <li>○ Keil compiler</li> <li>○ BiPOM ELECTRONIC</li> <li>○ Green Hill software</li> </ul> </li> </ul>
USABILITY AND APPLICATION	<ul style="list-style-type: none"> <li>• C language has a free-format of program coding.</li> <li>• It is specifically used for desktop application.</li> <li>• Optimization is normal.</li> <li>• It is very easy to read and modify the C language.</li> <li>• Bug fixing are very easy in a C language program.</li> <li>• It supports other various programming languages during application.</li> <li>• Input can be given to the program while it is running.</li> <li>• Applications of C Program: <ul style="list-style-type: none"> <li>○ Logical programs</li> <li>○ System software programs</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Formatting depends upon the type of microprocessor that is used.</li> <li>• It is used for limited resources like RAM and ROM.</li> <li>• High level of optimization.</li> <li>• It is not easy to read and modify the Embedded C language.</li> <li>• Bug fixing is complicated in a Embedded C language program.</li> <li>• It supports only required processor of the application, and not the programming languages.</li> <li>• Only the pre-defined input can be given to the running program.</li> <li>• Applications of Embedded C Program: <ul style="list-style-type: none"> <li>○ DVD</li> <li>○ TV</li> <li>○ Digital camera</li> </ul> </li> </ul>

## **MISRA:**

MISRA (the Motor Industry Software Reliability Association) provides guidelines for developing safety- and security-related electronic systems, embedded control systems, software-intensive applications, and standalone software.

MISRA is a collaborative effort among vehicle manufacturers, component suppliers, and engineering consultancies. It is managed by a Steering Committee that includes Ford Motor Company, Bentley Motors, Jaguar Land Rover, HORIBA MIRA, ZF TRW, and the University of Leeds.

Though born in the automotive industry, MISRA has gained acceptance in other markets such as aerospace, biomedical, and financial. It is accepted across embedded, IoT, and industrial control systems as well. While MISRA compliance doesn't guarantee that software will be free from all quality or security issues, it does produce code that is more robust, easier to maintain, and more portable.

## **MISRA CODING STANDARD:**

- The C programming language is one of the most popular languages for embedded systems because of its intrinsic capabilities, such as performance, portability across hardware, and direct control on memory. However, certain C language constructs can lead to programming errors, undefined behavior, or implementation-defined behavior.
- The MISRA C guidelines define a “safe-subset” of the C language to protect against language aspects that can compromise the safety and security of embedded systems.
- MISRA provides coding standards for developing safety-critical systems.
- MISRA is made up of manufacturers, component suppliers, and engineering consultancies. Experts from Perforce's static code analysis team (formerly PRQA) are members of MISRA, too.
- MISRA first developed coding guidelines in 1998. These were specific to the C programming language. Since then, MISRA has added a coding standard for C++.

## **EVOLUTION OF MISRA:**

The first edition of MISRA C was published in 1998. MISRA has since published two more editions, MISRA C:2004 and MISRA C:2012, and two amendments to MISRA C:2012. With each publication, MISRA has added new guidelines and provided more direction on how to achieve MISRA C compliance.

MISRA C:1998	MISRA C:2004	MISRA C:2012	MISRA C:2012 Amendment 1	MISRA C:2012 Amendment 2
Published April 1998	Published October 2004	Published March 2013	Published April 2016	Published February 2020
Guidelines 127	Guidelines 142	Guidelines 169	Guidelines 173	Guidelines 175
Language C90	Language C90	Languages C90 and C99	Languages C90 and C99	Languages C90, C99, C11 and C18

## HOW TO ACHIEVE MISRA COMPLIANCE?

Achieving MISRA compliance takes knowledge, skill, and the right tools.

Here are seven steps to comply with MISRA:

1. **Know the Rules:** You need to know the MISRA coding rules pertinent to which version of C or C++ you're using.
2. **Check Your Code Constantly:** Continuously inspecting your code for violations is the best way to improve quality.
3. **Set Baselines:** Embedded systems come with legacy codebases. By setting baselines, you can focus on making sure your new code is compliant.
4. **Prioritize Violations Based on Risk:** You could have hundreds or even thousands of violations in your code. That's why it's important to prioritize rule violations based on risk severity. Some static code analysis tools can do this for you.
5. **Document Your Deviations:** Sometimes there are exceptions to the rule. But when it comes to compliance, every rule deviation needs to be well-documented.
6. **Monitor Your MISRA Compliance:** Keep an eye on how MISRA compliant your code is. Using a static code analyzer makes this easier by automatically generating a compliance report.
7. **Choose the Right Static Code Analyzer:** Choosing the right static code analyzer makes everything else easy. It takes care of scanning your code — new and legacy — for violations. It prioritizes vulnerabilities based on risk.

## WHAT ARE MISRA C RULES?

- **Rule 21.21:** The Standard Library function system of `<stdlib.h>` shall not be used
- **Rule 12.5:** The `*sizeof*` operator shall not have an operand which is a function parameter declared as "array of type".
- **Rule 18.1:** A pointer resulting from arithmetic on a pointer operand shall address an element of the same array as that pointer operand
- **Rule 14.9:** An `if (expression)` construct shall be followed by a compound statement. The `else` keyword shall be followed by either a compound statement, or another `if` statement.

- **Rule 14.10:** All if ... else if constructs shall be terminated with an else clause.
- **Rule 59:** The statement forming the body of an "if", "else if", "else", "while", "do ... while", or "for" statement shall always be enclosed in braces
- **Rule 1.5:** Obsolescent language features shall not be used
- **Rule 6.3:** Bit field in unions
- **Rule 7.5:** Integer-constant macros
- **Rule 18.9:** Object lifetime
- **Rules 21.22-21.23:** Type generic math macros (<tgmath.h>)
- **Rule 21.24:** The random number generator functions of <stdlib.h>
- **Directive 4.15:** Floating point (including comparisons, NaNs, and infinities)
- **Rules 8.15-8.17:** Alignment of objects (<stdalign.h>)
- **Rules 17.9-17.13:** No-return functions (<stdnoreturn.h>)
- **Rules 23.1-23.7:** Type generic expressions (\_Generic)
- **Rule 1.4:** Emergent language features shall not be used
- **Rule 13.2:** The value of an expression and its persistent side effects shall be the same under all permitted evaluation orders
- **Rule 17.2:** Functions shall not call themselves, either directly or indirectly - **Rule 19.2:** The union keyword should not be used

## WHAT IS CERT STANDARD:

*CERT is a secure coding standard that supports commonly used programming languages such as C, C++, and Java.*

The standards are developed through a broad-based community effort by members of the software development and software security communities.

The rules and recommendations target insecure coding practices and undefined behaviors that lead to security risks.

The latest rules and recommendations are available on the secure coding standard's website and are also periodically published: C and C++ in 2016 and Java in 2011.

## CERT CODING STANDARD:

Some of the coding standard followed in CERT C are

- 1) The aim of the secure coding standard is to not only detect security risks with rules but also provide suggestions that can improve code quality with recommendations.



- 2) The scope of the secure coding standard is a whole program coding standard and the goal is to produce safe, reliable, and secure systems.
- 3) By using the secure coding standard, you are able to ensure that your software is secure and safeguarded from potential vulnerabilities.

## WHAT ARE CERT C RULES?

### 1) Preprocessor (PRE)

- PRE30-C. Do not create a universal character name through concatenation
- PRE31-C. Avoid side effects in arguments to unsafe macros
- PRE32-C. Do not use preprocessor directives in invocations of function-like macros 2)

### Declarations and Initialization (DCL)

- DCL30-C. Declare objects with appropriate storage durations
- DCL31-C. Declare identifiers before using them
- DCL36-C. Do not declare an identifier with conflicting linkage classifications
- DCL37-C. Do not declare or define a reserved identifier
- DCL38-C. Use the correct syntax when declaring a flexible array member
- DCL39-C. Avoid information leakage when passing a structure across a trust boundary
- DCL40-C. Do not create incompatible declarations of the same function or object
- DCL41-C. Do not declare variables inside a switch statement before the first case label

### 3) Expressions (EXP)

- EXP33-C. Do not read uninitialized memory
- EXP34-C. Do not dereference null pointers
- EXP35-C. Do not modify objects with temporary lifetime 4) Arrays (ARR)

- ARR30-C. Do not form or use out-of-bounds pointers or array subscripts
- ARR32-C. Ensure size arguments for variable length arrays are in a valid range
- ARR36-C. Do not subtract or compare two pointers that do not refer to the same array
- ARR37-C. Do not add or subtract an integer to a pointer to a non-array object
- ARR38-C. Guarantee that library functions do not form invalid pointers
- ARR39-C. Do not add or subtract a scaled integer to a pointer 5) Input Output (FIO)

- FIO30-C. Exclude user input from format strings
- FIO32-C. Do not perform operations on devices that are only appropriate for files
- FIO34-C. Distinguish between characters read from a file and EOF or WEOF

- FIO37-C. Do not assume that `fgets()` or `fgetws()` returns a nonempty string when successful
- FIO38-C. Do not copy a `FILE` object

#### 6) Error Handling (ERR)

- ERR30-C. Take care when reading `errno`
- ERR32-C. Do not rely on indeterminate values of `errno`
- ERR33-C. Detect and handle standard library errors
- ERR34-C. Detect errors when converting a string to a number

#### **RESULT:**

Thus, a study experiment on Embedded C coding standard is made successfully.

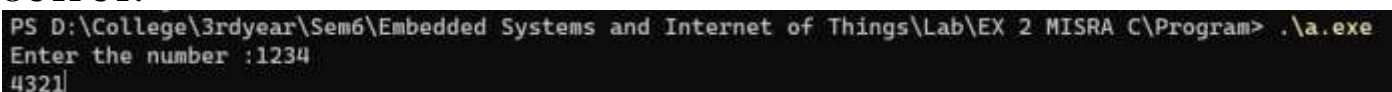
EX.NO: 2	<b>DEMONSTRATION OF MISRA C CODING STANDARD</b>
DATE:20/12/22	

**AIM:**

To analyse various C programs and to demonstrate them in MISRA C coding standard.

**REVERSING A NUMBER****C PROGRAM:**

```
#include <stdio.h>
#include <conio.h> void
main()
{   int num, rev = 0;
printf("Enter the number :");
scanf("%d", &num);   while
(num > 0)
{
    rev = (rev * 10) + (num % 10);
num = num / 10;
}   printf("%d",
rev);   getch();
}
```

**OUTPUT:**

```
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program> .\a.exe
Enter the number :1234
4321|
```

**MISRA C STANDARD:**

**Rule 1.1(required):** All code shall confirm to ISO/IEC 9898:1990

**Line #1:**

Rule 19.1(A):#include statement in a file should only be preceded by other pre-processor or directories or comments.

The above rule has been adapted in the coding practice.

*#include<stdio.h>*

**Line#3:**

Rule 16.1(R): Function shall not be defined with variable number of arguments.

Rule 16.2(R): Function shall not call themselves either directly or indirectly.

The above rule has been adapted in the coding practice. *int*

*main()*

**Line#5:**

Rule 6.2(R): Unsigned character type shall be used only for numeric value.

The above rule has been adapted in the coding practice. *unsigned*

*int num, rev=0;*

**Line#6:**

Rule 4.1(R): Only escape sequences that are defined in ISO standard are permitted. All hexadecimal escape sequences are not permitted.

The above rule has been adapted in the coding practice.

*printf("\n\n Enter the number:\n");* **Line#7:**

Rule 16.1(R): Function shall not be defined with variable number of arguments.

Rule 16.2(R): Function shall not call themselves either directly or indirectly.

The above rule has been adapted in the coding practice.

*scanf("%d",&n);*

**MISRA C PROGRAM:**

```
#include "stdio.h"
```

```
#include "conio.h" void
```

```
main()
```

```
{
```

```
    unsigned int num, rev = 0;
```

```
    printf("\nEnter the number:\n");
```

```
    scanf("%d", &num);    while
```

```
(num > 0)
```

```
    {
```

```
        rev = (rev * 10) + (num % 10);
```

```
    num = num / 10;
```

```
    }
```

```
    printf("\n%d\n", rev);
```

```
    getch();
```

```
}
```

## OUTPUT:

```
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program\MISRA C> gcc .\ReverseNumbers.c
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program\MISRA C> .\a.exe

Enter the number:
453
354
```

## SWAP TWO NUMBERS C

### PROGRAM:

```
#include <stdio.h>
#include <conio.h> void
swap(int a, int b)
{   int temp;   temp = a;   a = b;   b =
temp;   printf("After swapping : %d
%d", a, b);
} int main() {   int num1, num2;
printf("Enter two numbers : ");
scanf("%d %d", &num1, &num2);
swap(num1, num2);   getch();
}
```

## OUTPUT:

```
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program> .\a.exe
Enter two numbers : 30 11
After swapping : 11 30
```

## MISRA C CODING STANDARD:

**Rule 1.1(required):** All code shall confirm to ISO/IEC 9898:1990

### Line #1:

Rule 19.1(A):#include statement in a file should only be preceded by other preprocessor or directories or comments.

The above rule has been adapted in the coding practice.

*#include<stdio.h>*

### Line #3:

Rule 16.1(R): Function shall not be defined with variable number of arguments.

The above rule has been adapted in the coding practice. *void swap (int a, int b);*

**Line#11:**

Rule 16.1(R): Function shall not be defined with variable number of arguments.

Rule 16.2(R): Function shall not call themselves either directly or indirectly.

The above rule has been adapted in the coding practice. *void*

*main()*

**Line#13:**

Rule 6.2(R): Unsigned character type shall be used only for numeric value.

The above rule has been adapted in the coding practice. *unsigned*

*int n;*

**Line#15:**

Rule 4.1(R): Only escape sequences that are defined in ISO standard are permitted. All hexadecimal escape sequences are not permitted.

The above rule has to be adapted in the coding practice.

*printf("\n Enter two numbers:\n");*

**Line#16:**

Rule 16.1(R): Function shall not be defined with variable number of arguments.

Rule 16.2(R): Function shall not call themselves either directly or indirectly.

The above rule has been adapted in the coding practice.

*scanf("%d %d", &a, &b);*

**MISRA C PROGRAM:**

```
#include "stdio.h"
```

```
#include "conio.h" void
```

```
swap(int a, int b)
```

```
{
```

```
    unsigned int temp;
```

```
    temp = a;    a = b;
```

```
    b = temp;
```

```
    printf("\nAfter swapping: %d %d\n", a, b);
```

```
} int
```

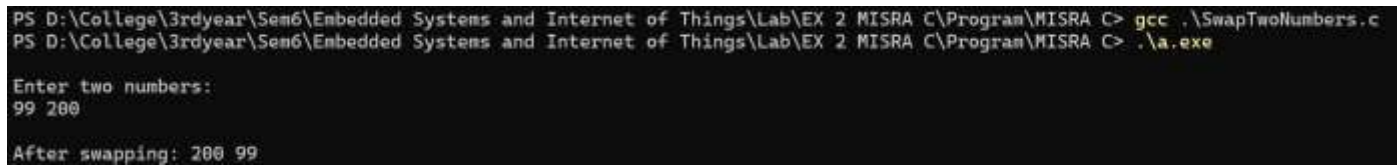
```
main()
```

```
{
```

```
    int num1, num2;
```

```
printf("\nEnter two numbers:\n");  
scanf("%d %d", &num1, &num2);  
swap(num1, num2);  getch();  
}
```

### OUTPUT:



```
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program\MISRA C> gcc .\SwapTwoNumbers.c  
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 2 MISRA C\Program\MISRA C> .\a.exe  
  
Enter two numbers:  
99 200  
  
After swapping: 200 99
```

### RESULT:

Thus, various C programs have been analysed and they have been demonstrated in MISRA C coding standard.

EX. NO: 3	<b>DEMONSTRATION OF CERT C STANDARD</b>
DATE:27/12/22	

**AIM:**

To analyse various C programs and to demonstrate them in CERT C coding standard.

**FIBONACCI SERIES C****PROGRAM:**

```
#include <stdio.h>

int main() {
    int n, a, b, nt, i;
    printf("Enter a number:");
    scanf("%d", &n);    a = 0;
    b = 1;
    if (n > 0 && n <= 2)
    {
        printf("%d %d", a, b);
        return 0;
    }
    else {
        printf("%d %d", a, b);
        for (i = 2; i < n; i++)
        {
            nt = a + b;
            a = b;
            b = nt;
            printf(" %d", nt);
        }
        printf("\n");
        return 0;
    }
}
```

**OUTPUT:**

```
PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C> .\a.exe
Enter a number:5
0 1 1 2 3
```



## CERT C STANDARD:

### Line #1:

PRE04-C: Do not reuse a standard header file name

If a file with the same name as a standard file name is placed in the search path for included source files, the behaviour is undefined.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
PRE04-C	Low	Unlikely	Medium	P2	L3

PRE08-C: Guarantee header file names are unique

Guarantee header file names are unique, all included files should differ (in a case insensitive manner) in their first eight characters or in their (one character) file extension.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
PRE08-C	Low	Unlikely	Medium	P2	L3

```
#include<stdio.h>
```

### Line #4:

DCL02-C: Use visually distinct identifiers

Use visually distinct identifiers to eliminate errors resulting from mis recognizing the spelling of an identifier during the development and review of code.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL02-C	Low	Unlikely	Medium	P2	L3

DCL04-C: Take care when declaring more than one variable per declaration

Declaring multiple variables in a single declaration can cause confusion regarding the types of the variables and their initial values. If more than one variable is declared in a declaration, care must be taken that the actual type and initialized value of the variable is known.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL04-C	Low	Unlikely	Low	P3	L3

```
int n, a, b, nt, i;
```

**Line #7:**

DCL00-C: Declare immutable values using const or enum.

Immutable (constant values) should be declared as const-qualified objects (unmodifiable lvalues), enumerations values, or as a last resort, a #define.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL00-C	Low	Unlikely	High	P1	L3

DCL06-A: Use meaningful symbolic constants to represent literal values in program logic

Avoid the use of magic numbers in code when possible. Magic numbers are constant values that represent an arbitrary value, such as a determined appropriate buffer size.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL06-C	Low	Unlikely	Medium	P2	L3

*a = 0;*

*b = 1;*

**Line #17:**

INT01-A: Use size\_t for all integer values representing the size of an object.

The size\_t type is the unsigned integer type of the result of the sizeof operator. The underlying representation of variables of type size\_t are guaranteed to be of sufficient precision to represent the size of an object.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
INT01-C	Medium	Probable	Medium	P8	L2

*for (i = 2; i < n; i++)*

**CERT C PROGRAM:**

```
#include <stddef.h>
#include <stdio.h>
#define SIZE 7 unsigned
int main()
{   unsigned int a = 0;
    unsigned int b = 1;   unsigned
    int nt;   if (SIZE > 0 &&
```

```

SIZE <= 2)    {
printf("%d %d", a, b);
return 0;
    }
else
    {
        printf("%d %d", a, b);
for (size_t i = 2; i < SIZE; i++)
    {
        nt = a + b;
a = b;        b = nt;
printf(" %d", nt);
    }    }
printf("\n");
return 0;
}

```

## OUTPUT:

```

D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C\Program\CERT C>gcc Fibonacci.c
D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C\Program\CERT C>a.exe
0 1 1 2 3 5 8

```

## ADDITION OF TWO-DIMENSIONAL ARRAYS C

### PROGRAM:

```

#include <stdio.h>
int main()
{
    int n, i, j;
    int a[100][100], b[100][100];
printf("Enter array size\n");
scanf("%d", &n);
printf("Enter array 1 elements\n");
for (i = 0; i < n; i++)
    {
for (j = 0; j < n; j++)
{
        scanf("%d", &a[i][j]);

```

```

    }
}
printf("Enter array 2 elements\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < n;
j++)
    {
        scanf("%d", &b[i][j]);
    }
}
printf("Sum of array elements:\n");
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        printf("%d ", a[i][j] + b[i][j]);
    }
    printf("\n");
}
return 0;
}

```

## OUTPUT:

```

PS D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C\Program> .\a.exe
Enter array size
2
Enter array 1 elements
1 2 3 4
Enter array 2 elements
1 2 3 4
Sum of array elements:
2 4
6 8

```

## CERT C STANDARD:

### Line #1:

PRE04-C: Do not reuse a standard header file name

If a file with the same name as a standard file name is placed in the search path for included source files, the behaviour is undefined.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
PRE04-C	Low	Unlikely	Medium	P2	L3

PRE08-C: Guarantee header file names are unique

Guarantee header file names are unique, all included files should differ (in a case insensitive manner) in their first eight characters or in their (one character) file extension.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
PRE08-C	Low	Unlikely	Medium	P2	L3

*#include<stdio.h>*

#### Line #4:

DCL02-C: Use visually distinct identifiers

Use visually distinct identifiers to eliminate errors resulting from mis recognizing the spelling of an identifier during the development and review of code.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL02-C	Low	Unlikely	Medium	P2	L3

*int n, i, j;*

#### Line #5:

DCL00-C: Declare immutable values using const or enum.

Immutable (constant values) should be declared as const-qualified objects (unmodifiable lvalues), enumerations values, or as a last resort, a #define.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL00-C	Low	Unlikely	High	P1	L3

DCL06-C: Use meaningful symbolic constants to represent literal values in program logic

Avoid the use of magic numbers in code when possible. Magic numbers are constant values that represent an arbitrary value, such as a determined appropriate buffer size.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
DCL06-C	Low	Unlikely	Medium	P2	L3

*int a[100][100], b[100][100];*

**Line #9:**

INT01-C: Use size\_t for all integer values representing the size of an object.

The size\_t type is the unsigned integer type of the result of the sizeof operator. The underlying representation of variables of type size\_t are guaranteed to be of sufficient precision to represent the size of an object.

Recommendation	Severity	Likelihood	Remediation Cost	Priority	Level
INT01-C	Medium	Probable	Medium	P8	L2

*for (i = 0; i < n; i++)*

**CERT C PROGRAM:**

```
#include <stddef.h>
#include <stdio.h>
#define SIZE 3 static int
a[SIZE][SIZE]; static int
b[SIZE][SIZE];
unsigned int main()
{   unsigned int x = 1;   for
(size_t i = 0; i < SIZE; i++)
    {
        for (size_t j = 0; j < SIZE; j++)
        {
a[i][j] = x;
b[i][j] = x;
x++;
        }
    }
    printf("Sum of array elements:\n");
for (size_t i = 0; i < SIZE; i++)
    {   for (size_t j = 0; j < SIZE;
j++)
        {   printf("%d ", a[i][j] +
b[i][j]);
        }
    }
printf("\n");
```

```
    }  
    return 0;  
}
```

## OUTPUT:

```
D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C\Program\CERT C>gcc 2DArrayAddition.c  
D:\College\3rdyear\Sem6\Embedded Systems and Internet of Things\Lab\EX 3 CERT C\Program\CERT C>a.exe  
Sum of array elements:  
2 4 6  
8 10 12  
14 16 18
```

## RESULT:

Thus, various C programs have been analysed and they have been demonstrated in CERT C coding standard.

## LINUX INSTALLATION PROCEDURE

**AIM:**

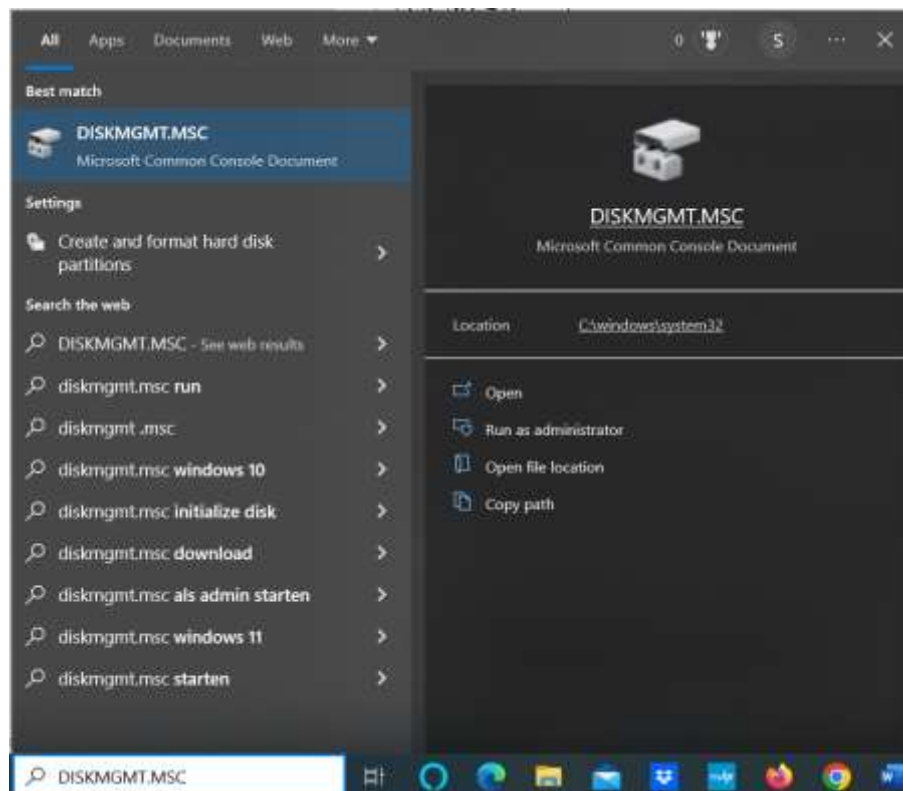
To install and analyse Linux on Windows Operating System.

**LINUX:**

Linux is a family of open-source operating systems. They are based on the Linux kernel and are free to download. They can be installed on either a Mac or Windows computer. If you want to dual boot Linux and Windows, you will need to create a space for your Linux OS to live. In order to do this, you will have to partition your main hard drive.

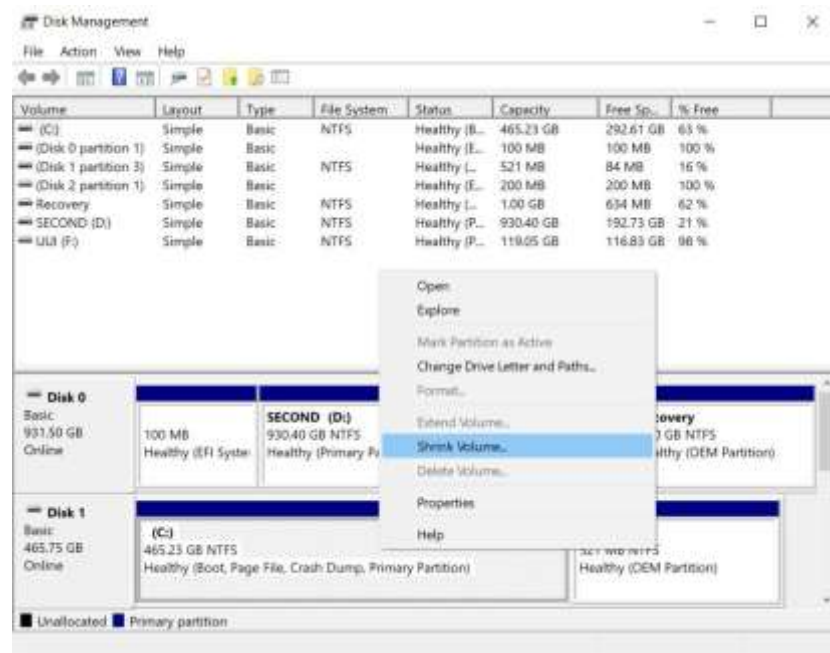
**PROCEDURE:****Partition a Hard Drive in Windows 10:**

- ✓ Open the Windows Search Bar. This is the magnifying glass-shaped icon in the bottom-left corner of your screen.
- ✓ Then type “DISKMGMT.MSC” in the search bar and hit enter.

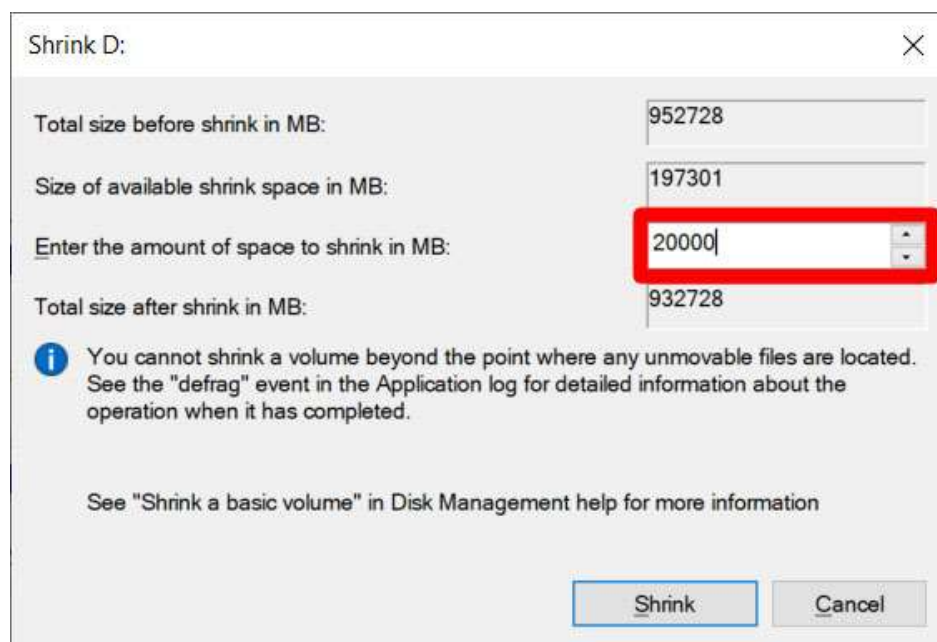




- ✓ Right-click on your main hard drive and select Shrink Volume. If you have more than one drive, make sure to choose the one that says Primary Partition. This will usually be labeled as the C: drive.



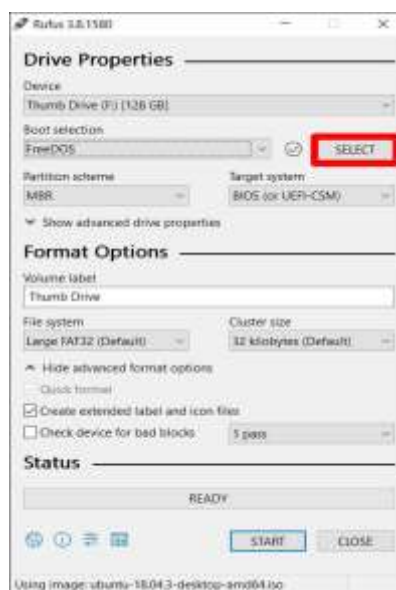
- ✓ Then choose how much you want to shrink your drive. It is recommended that you set aside at least 20GB (20,000MB) for Linux.



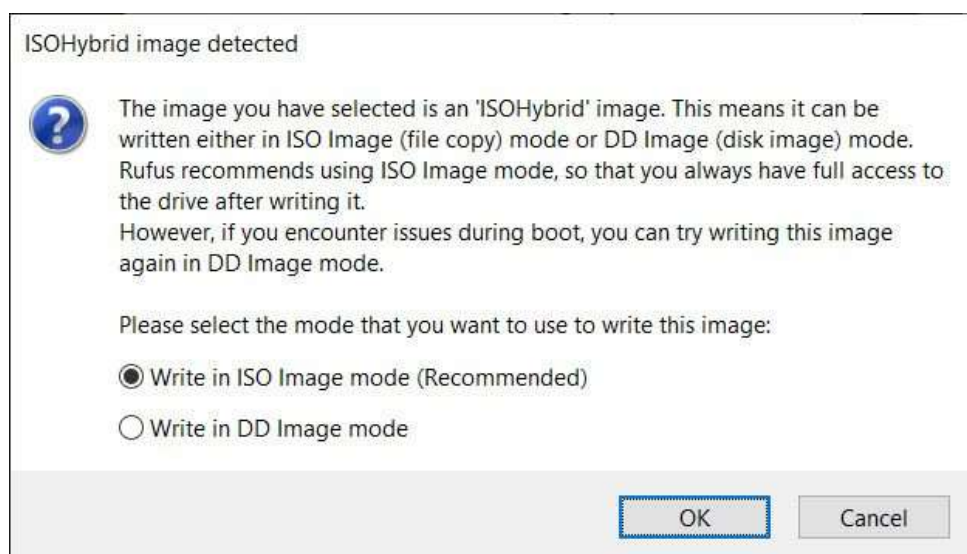
- ✓ Finally, click Shrink. Once you have a designated space to install Linux, you'll need to write a Linux Distro onto a USB thumb drive or external drive 4GB or larger.

## Make a Linux Bootable USB:

- ✓ Download a Linux distro in ISO format. An ISO file is a disk image. Some of the top options are Ubuntu, Mint, or Fedora. They are free to download from each distribution's main website. For this article, we are using Ubuntu.
- ✓ Insert the USB drive into your computer. You might be asked to format your drive. This will erase all the data stored on your drive, so make sure to back up your files before you begin.
- ✓ Download Rufus. You can find the latest version of the application [here](#).
- ✓ Open Rufus and select your USB drive from the Device list. If you don't know which drive to use, eject all other drives until you only have one to choose from.
- ✓ Under Boot Selection, click the Select button and choose the ISO file you downloaded earlier. Don't change the other default settings.

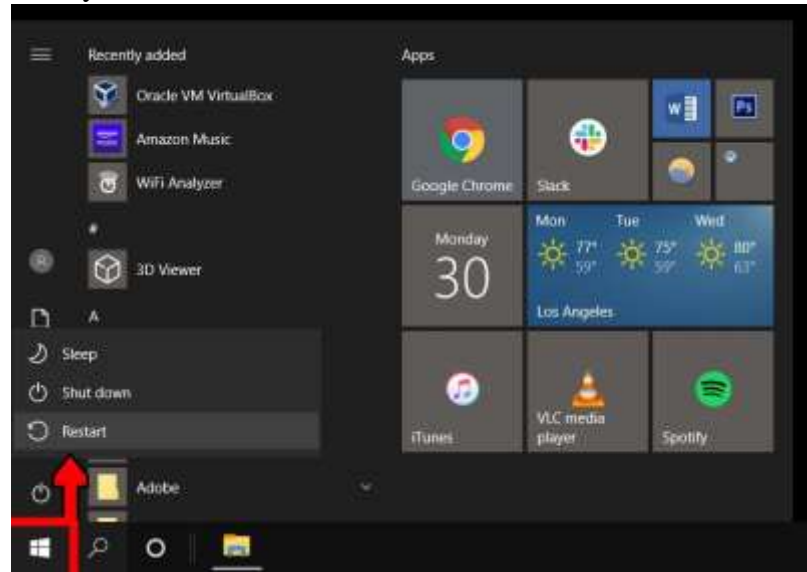


- ✓ Finally, click Start. If you get a pop-up message asking you to select a mode that you want to use to write the image, choose ISO.

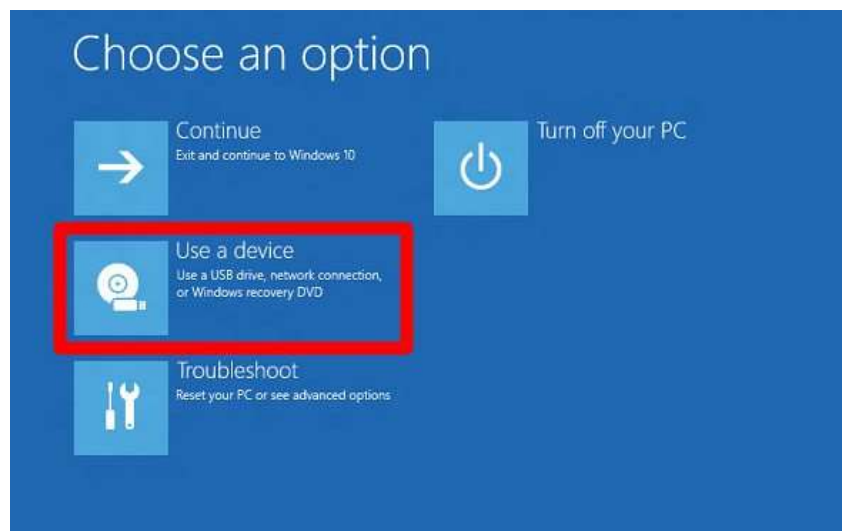


## Install Linux from USB:

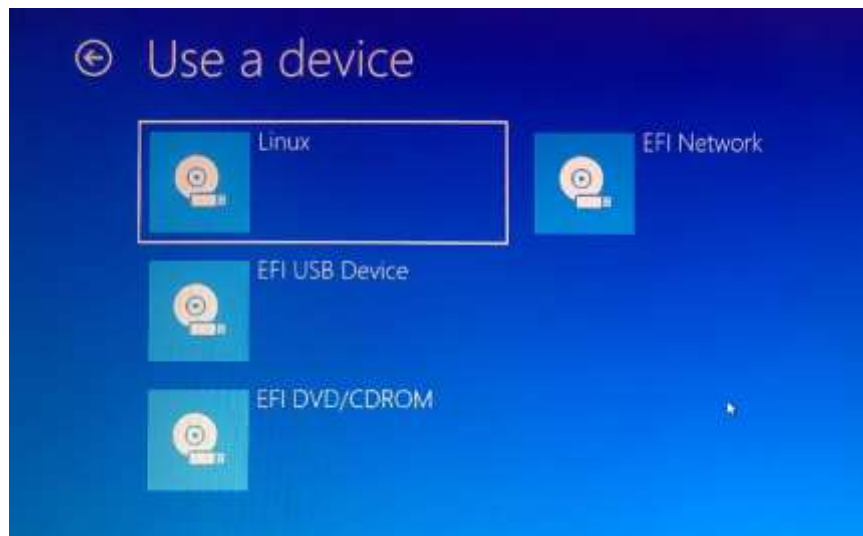
- ✓ Now that you have your Linux distro on a USB, here's how to
- ✓ Insert a bootable Linux USB drive.
- ✓ Click the start menu. This is the button in the lower-left corner of your screen that looks like the Windows logo.
- ✓ Then hold down the **SHIFT** key while clicking Restart. This will take you into the Windows Recovery Environment.



- ✓ Then select Use a Device.



- ✓ Find your device in the list. If you don't see your drive, choose EFI USB Device, then pick your drive from the next screen.

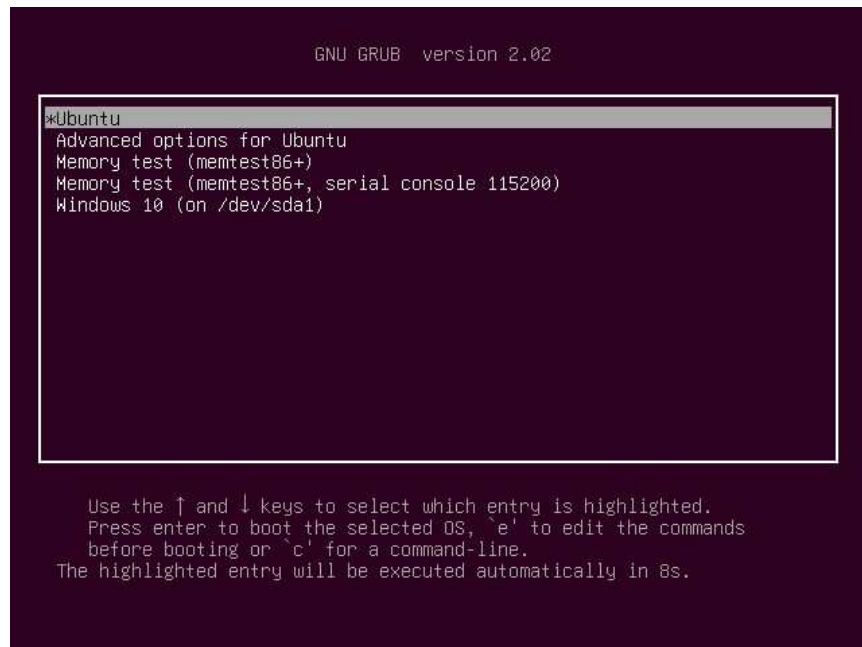


- ✓ Your computer will now boot Linux. If your computer reboots Windows, there was either an issue with your drive, or you might have to change settings in your BIOS.
- ✓ Warning: Changing BIOS settings can damage your computer if you don't know what you're doing.
- ✓ Select Install Linux. Some distros also let you try out the OS before installing it here.



- ✓ Go through the installation process. This will differ depending on which distro you are trying to install. These details might include your WiFi network, language, time zone, keyboard layout, etc. You might also be required to create an account with a username and password. Make sure to write down any details, as you will likely need them in the future.
- ✓ Most distros will allow you to partition your drive or erase it and do a clean install during the installation.
- ✓ Warning: Erasing your disk will mean you will lose your settings, files, and Windows operating system. Only select Erase if you have saved copies of all your files before starting the install process.

- ✓ Reboot your computer when prompted. If you have more than one OS in your system, you will be taken to a GNU GRUB screen after rebooting. This screen allows you to select which OS you want to boot.



## RESULT:

Thus, installation of Linux on Windows Operating System is executed and output verified successfully.

EX. NO: 4 B

DATE:03/01/23

## LINUX INSTALLATION PROCEDURE

### AIM:

To upgrade the kernel file of the Linux operating system using kernel compilation process.

### APPARATUS REQUIRED:

- ✓ Linux OS
- ✓ Virtual Machine

### DESCRIPTION:

Linux kernel compilation provides the user to unlock the features which are not available for the standard users. Linux compilation process allows the users to modify their kernel depending on their hardware and software application environment. Following points shows the steps involved in compiling a kernel.

### Building Linux Kernel

The process of building a Linux kernel can be performed in seven easy steps. However, the procedure may require a significant amount of time to complete, depending on the system speed.

**Follow the steps below to build the latest Linux kernel.**

Step 1: Download the Source Code Visit the official kernel website and download the latest kernel version. The downloaded file contains a compressed source code.





Open the terminal and use the wget command to download the Linux kernel source code: `wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz`

The output shows the “saved” message when the download completes.

```
ubuntu@ubuntu: ~$ wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
--2023-01-09 07:13:58-- https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.0.7.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 199.232.253.176, 2a04:4e42:fd3::432
Connecting to cdn.kernel.org (cdn.kernel.org)|199.232.253.176|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 133884956 (128M) [application/x-xz]
Saving to: 'linux-6.0.7.tar.xz'

linux-6.0.7.tar.xz 100%[=====] 127.68M 1.37MB/s in 19m 20s
2023-01-09 07:33:21 (113 KB/s) - 'linux-6.0.7.tar.xz' saved [133884956/133884956]
```

## Step 2: Extract the Source Code

When the file is ready, [run the tar command](#) to extract the source code:

`tar xvf linux-6.0.7.tar.xz` The output displays the extracted kernel source code:

```
ubuntu@ubuntu:~$ tar xvf linux-6.0.7.tar.xz
linux-6.0.7/virt/
linux-6.0.7/virt/Makefile
linux-6.0.7/virt/kvm/
linux-6.0.7/virt/kvm/Kconfig
linux-6.0.7/virt/kvm/Makefile.kvm
linux-6.0.7/virt/kvm/async_pf.c
linux-6.0.7/virt/kvm/async_pf.h
linux-6.0.7/virt/kvm/binary_stats.c
linux-6.0.7/virt/kvm/coalesced_mmio.c
linux-6.0.7/virt/kvm/coalesced_mmio.h
linux-6.0.7/virt/kvm/dirty_ring.c
linux-6.0.7/virt/kvm/eventfd.c
linux-6.0.7/virt/kvm/irqchip.c
linux-6.0.7/virt/kvm/kvm_main.c
linux-6.0.7/virt/kvm/kvm_mm.h
linux-6.0.7/virt/kvm/pfncache.c
linux-6.0.7/virt/kvm/vfio.c
linux-6.0.7/virt/kvm/vfio.h
linux-6.0.7/virt/lib/
linux-6.0.7/virt/lib/Kconfig
linux-6.0.7/virt/lib/Makefile
linux-6.0.7/virt/lib/irqbypass.c
```

## Step 3: Install Required Packages

Install additional packages before building a kernel. To do so, run this command:

`sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison`

The command we used above installs the essential packages for performing Linux kernel compilation

```
ubuntu@ubuntu:~/linux-6.0.7$ sudo apt-get -f install
[sudo] password for ubuntu:
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 285 not upgraded.
ubuntu@ubuntu:~/linux-6.0.7$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu cpp-9 dpkg-dev fakeroot
  g++ g++-9 gcc gcc-9 gcc-9-base libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
  libbinutils libc-dev-bin libc6 libc6-dbg libc6-dev libcrypt-dev
  libctf-nobfd0 libctf0 libdpkg-perl libfakeroot libgcc-9-dev libitm1 liblsan0
  libquadmath0 libstdc++-9-dev libtsan0 libubsan1 linux-libc-dev make
  manpages-dev
```

```
ubuntu@ubuntu:~/linux-6.0.7$ sudo apt-get install git fakeroot build-essential ncurses-dev xz-utils libssl-dev bc flex libelf-dev bison
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libncurses-dev' instead of 'ncurses-dev'
bc is already the newest version (1.07.1-2build1).
bc set to manually installed.
fakeroot is already the newest version (1.24-1).
fakeroot set to manually installed.
build-essential is already the newest version (12.8ubuntu1.1).
The following additional packages will be installed:
  git-man liberror-perl libfl-dev libfl2 libsigsegv2 libssl1.1 m4 zlib1g
  zlib1g-dev
Suggested packages:
  bison-doc flex-doc git-daemon-run | git-daemon-sysvinit git-doc git-el
  git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn ncurses-doc
  libssl-doc m4-doc
The following NEW packages will be installed:
  bison flex git git-man libelf-dev liberror-perl libfl-dev libfl2
  libncurses-dev libsigsegv2 libssl-dev m4 zlib1g-dev
The following packages will be upgraded:
```

#### Step 4: Configure Kernel

The Linux kernel source code comes with the default configuration. However, you can adjust it to your needs. To do so, follow the steps below:

Navigate to the linux-6.0.7 directory using the cd command:

```
cd linux-6.0.7
```

Copy the existing configuration file using the cp command:

```
cp -v /boot/config-$(uname -r) .config
```

```
manko@pnap:~$ cd linux-6.0.7/
manko@pnap:~/linux-6.0.7$ cp -v /boot/config-$(uname -r) .config
'/boot/config-5.15.0-52-generic' -> '.config'
manko@pnap:~/linux-6.0.7$
```

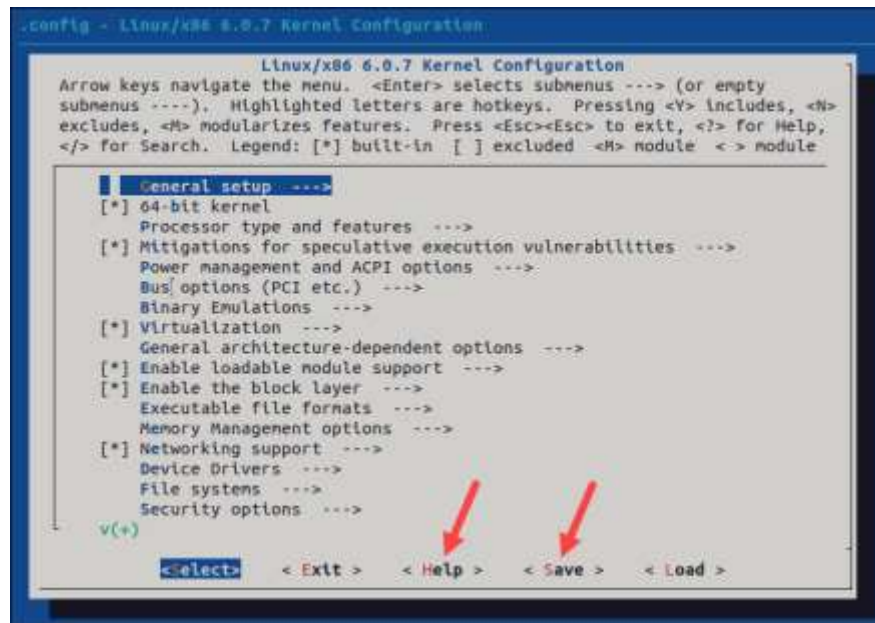
To make changes to the configuration file, run the make command:

make menuconfig, this command launches several scripts that open the configuration menu:

```
ubuntu@ubuntu:~/linux-6.0.7$ make menuconfig
UPD      scripts/kconfig/mconf-cfg
HOSTCC   scripts/kconfig/mconf.o
HOSTCC   scripts/kconfig/ldialog/checklist.o
HOSTCC   scripts/kconfig/ldialog/inputbox.o
HOSTCC   scripts/kconfig/ldialog/menubox.o
HOSTCC   scripts/kconfig/ldialog/textbox.o
HOSTCC   scripts/kconfig/ldialog/util.o
HOSTCC   scripts/kconfig/ldialog/yesno.o
HOSTCC   scripts/kconfig/confdata.o
HOSTCC   scripts/kconfig/expr.o
LEX       scripts/kconfig/lexer.lex.c
YACC      scripts/kconfig/parser.tab.[ch]
HOSTCC   scripts/kconfig/lexer.lex.o
HOSTCC   scripts/kconfig/menu.o
HOSTCC   scripts/kconfig/parser.tab.o
HOSTCC   scripts/kconfig/preprocess.o
HOSTCC   scripts/kconfig/symbol.o
HOSTCC   scripts/kconfig/util.o
HOSTLD   scripts/kconfig/mconf
.config:434:warning: symbol value 'm' invalid for I8K
.config:1998:warning: symbol value 'm' invalid for MCTP
.config:8858:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC30
.config:8859:warning: symbol value 'm' invalid for VIDEO_ZORAN_ZR36060
.config:8860:warning: symbol value 'n' invalid for VIDEO_ZORAN_BUZ
.config:8861:warning: symbol value 'n' invalid for VIDEO_ZORAN_DC18
.config:8862:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33
.config:8863:warning: symbol value 'n' invalid for VIDEO_ZORAN_LML33R10
.config:8864:warning: symbol value 'n' invalid for VIDEO_ZORAN_AV56EYES
.config:9955:warning: symbol value 'n' invalid for ANDROID_BINDER_IPC
.config:9956:warning: symbol value 'n' invalid for ANDROID_BINDERFS
configuration written to .config
```



The configuration menu includes options such as firmware, file system, network, and memory settings. Use the arrows to make a selection or choose Help to learn more about the options. When you finish making the changes, select Save, and then exit the menu.



#### Step 5: Build the Kernel

Start building the kernel by running the following command: make

The process of building and compiling the Linux kernel takes some time to complete.

The terminal lists all Linux kernel components: memory management, hardware device drivers, filesystem drivers, network drivers, and process management.

```
marko@pnep:~/linux-6.0.7$ make
SYNC      include/config/auto.conf.cmd
HOSTCC    scripts/kconfig/conf.o
HOSTLD    scripts/kconfig/conf
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_32.h
SYSHDR    arch/x86/include/generated/asm/unistd_32_ia32.h
SYSHDR    arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_64.h
HYPERCALLS arch/x86/include/generated/asm/xen-hypercalls.h
HOSTCC    arch/x86/tools/relocs_32.o
HOSTCC    arch/x86/tools/relocs_64.o
HOSTCC    arch/x86/tools/relocs_common.o
HOSTLD    arch/x86/tools/relocs
HOSTCC    scripts/genksyms/genksyms.o
YACC      scripts/genksyms/parse.tab.[ch]
```

If you are compiling the kernel on Ubuntu, you may receive the following error that interrupts the building process:

No rule to make target 'debian/canonical-certs.pem'

Disable the conflicting security certificates by executing the two commands below: scripts/config --disable SYSTEM\_TRUSTED\_KEYS

scripts/config --disable SYSTEM\_REVOCATION\_KEYS

The commands return no output. Start the building process again with make, and press Enter repeatedly to confirm the default options for the generation of new certificates.

Install the required modules with this command:

sudo make modules\_install

```
marko@pnep:~/linux-6.0.7$ sudo make modules_install
INSTALL sound/usb/line6/snd-usb-line6.ko
INSTALL sound/usb/line6/snd-usb-pod.ko
INSTALL sound/usb/line6/snd-usb-podhd.ko
INSTALL sound/usb/line6/snd-usb-toneport.ko
INSTALL sound/usb/line6/snd-usb-variix.ko
INSTALL sound/usb/misc/snd-ua101.ko
INSTALL sound/usb/snd-usb-audio.ko
INSTALL sound/usb/snd-usbmidi-lib.ko
INSTALL sound/usb/usx2y/snd-usb-us122l.ko
INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
INSTALL sound/x86/snd-hdmi-lpe-audio.ko
INSTALL sound/xen/snd_xen_front.ko
DEPMOD 6.0.7
marko@pnep:~/linux-6.0.7$
```

Finally, install the kernel by typing:

sudo make install The output shows done when finished:

```
marko@pnep:~/linux-6.0.7$ sudo make install
sh ./arch/x86/boot/install.sh 6.0.7 arch/x86/boot/bzImage \
    System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/apt-auto-removal 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/dkms 6.0.7 /boot/vmlinuz-6.0.7
* dkms: running auto installation service for kernel 6.0.7 [ OK ]
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 6.0.7 /boot/vmlinuz-6.0.7
update-initramfs: Generating /boot/initrd.img-6.0.7
run-parts: executing /etc/kernel/postinst.d/update-notifier 6.0.7 /boot/vmlinuz-6.0.7
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 6.0.7 /boot/vmlinuz-6.0.7
Sourcing file '/etc/default/grub'
Sourcing file '/etc/default/grub.d/init-select.cfg'
Generating grub configuration file ...
done
marko@pnep:~/linux-6.0.7$
```

## Step 6: Update the Bootloader (Optional)

The GRUB bootloader is the first program that runs when the system powers on.

The make install command performs this process automatically, but you can also do it manually.

Update the initramfs to the installed kernel version:

sudo update-initramfs -c -k 6.0.7

Update the GRUB bootloader with this command:

sudo update-grub

### Step 7: Reboot and Verify Kernel Version

When you complete the steps above, reboot the machine.

When the system boots up, verify the kernel version using the uname command:

```
uname -mrs
```

The terminal prints out the current Linux kernel version.

```
marko@pnep:~$ uname -mrs  
Linux 6.0.7 x86_64  
marko@pnep:~$
```

### RESULT:

Thus, the procedure was followed and the kernel file of Linux OS was updated.

EX. NO: 5

DATE:10/01/23

## DEMONSTRATION OF LINUX COMMANDS

### AIM:

To study and execute basic LINUX commands.

### COMMANDS:

1) **pwd**

**Description:** The pwd command is used to display the location of the current working directory.

**Syntax:** pwd

**Output:** pwd/home/images.

2) **cd**

**Description:** The cd command is used to change the current directory.

**Syntax:** cd <directory name>

**Output:** cd desktop  
~/desktop\$

3) **touch**

**Description:** The touch command is used to create empty files. We can create multiple empty files by executing it once.

**Syntax:** touch <file name>

**Output:** touch text1.txt

4) **rename**

**Description:** The rename command is used to rename files. It is useful for renaming a large group of files.

**Syntax:** rename 's/old-name/new-name/' files

**Output:** 's/text1.txt/text2.txt/'\*txt

5) **tac**

**Description:** The tac command is the reverse of cat command, as its name specified. It displays the file content in reverse order (from the last line).

**Syntax:** tac <file name>

**Output:** tac text1.txt 11 10 9

6) **su**

**Description:** The su command provides administrative access to another user. In other words, it allows access of the Linux shell to another user.

**Syntax:** su <user name>

**Output:** su admin  
Password:

7) **id**

**Description:** The id command is used to display the user ID (UID) and group ID (GID).

**Syntax:** id

**Output:** uid=2001(user) gid=2001(user) groups=2001(user).

8) **useradd**

**Description:** The useradd command is used to add or remove a user on a Linux server.

**Syntax:** useradd username

**Output:** useradd sudo  
[sudo] password for admin:

9) **groupadd**

**Description:** The groupadd command is used to create a user group.

**Syntax:** groupadd <group name>

**Output:** groupadd team1

10) **cut**

**Description:** The cut command is used to select a specific column of a file. The '-d' option is used as a delimiter and, the '-f' option is used to specify a column number.

**Syntax:** cut -d(delimiter) -f(columnNumber) <fileName>

**Output:** cut -d- -f2- text1.txt (It displays second column)

11) **wc**

**Description:** The wc command is used to count the lines, words, and characters in a file.

**Syntax:** wc <file name>

**Output:** wc marks.txt  
6 0 0 marks.txt

12) **od**

**Description:** The od command is used to display the content of a file in different s, such as hexadecimal, octal, and ASCII characters.

**Syntax:** od -b <fileName> // Octal format  
od -t x1 <fileName> // Hexa decimal format  
od -c <fileName> // ASCII character format

**Output:** od -b text1.txt

0000000 061 012 062 012 063 012 064 012 065 0000011

13) **sort**

**Description:** The sort command is used to sort files in alphabetical order.

**Syntax:** sort <file name>

**Output:** sort text1.txt

14) **gzip**

**Description:** The gzip command is used to truncate the file size. It is a compressing tool. It replaces the original file by the compressed file having '.gz' extension.

**Syntax:** gzip <file1> <file2> <file3>

**Output:** gzip lib.txt file2.txt

15) **gunzip**

**Description:** The gunzip command is used to decompress a file.

**Syntax:** gunzip <file1> <file2> <file3>

**Output:** gunzip lib.txt file2.txt

16) **find**

**Description:** The find command is used to find a particular file within a directory.

**Syntax:** find. -name "\*.pdf"

**Output:** find. -name "text1.txt"

17) **sleep**

**Description:** The sleep command is used to hold the terminal by the specified amount of time. By default, it takes time in seconds.

**Syntax:** sleep <time>

**Output:** sleep 5

18) **time**

**Description:** The time command is used to display the time to execute a command.

**Syntax:** time

**Output:** time

real 0m0.000s

user 0m0.000s

sys 0m0.000s

19) **zcat**

**Description:** The zcat command is used to display the compressed files.

**Syntax:** zcat <file name>

**Output:** zcat file1.txt

20) **exit**

**Description:** Linux exit command is used to exit from the current shell

**Syntax:** exit

**Output:** exit

21) **clear**

**Description:** Linux **clear** command is used to clear the terminal screen.

**Syntax:** clear

**Output:** After pressing the ENTER key, it will clear the terminal screen.

22) **ip**

**Description:** Linux ip command is an updated version of the ipconfig command. It is used to assign an IP address, initialize an interface, disable an interface.

**Syntax:** ip a or ip addr

**Output:** (It assign an IP address)

### 23) mail

**Description:** The mail command is used to send emails from the command line.

**Syntax:** mail -s "Subject" <recipient address>

**Output:** mail -s "Hello" maveen123@gmail.com

Cc:

HI, how are you...

### 24) host

**Description:** The host command is used to display the IP address for a given domain name and vice versa.

**Syntax:** host <domain name> or <ip address>

**Output:** host cocalc.com

cocalc.com has address 104.22.1.102

### 25) eject

**Description:** Eject command is used to Eject CD-ROM.

**Syntax:** eject /dev/cdrom

**Output:** (It will eject cd rom)

### 26) factor

**Description:** Factor command used to print prime factors.

**Syntax:** factor

**Output:** 25

25: 5 5

### 27) free

**Description:** Free command is used to display memory usage.

**Syntax:** free

**Output:** free

	total	used	free	shared	buff/cache
Mem:	32884788	864556	27552684	2816	4467548

### 28) kill

**Description:** Kill command is used to stop a process from running.

**Syntax:** kill

**Output:** kill

kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]

### 29) history

**Description:** The history command shows a list of the commands entered since you started the session.

**Syntax:** history

**Output:**

1 mkdir file

2 cat file1.txt

3 history

### 30) top

**Description:** The **top** command will display a list of running processes and how much CPU each process uses.

**Syntax:** top

**Output:** top

Tasks: 7 total, 2 running, 4 sleeping, 1 stopped, 0 zombie

%CPU(s): 0.6 us, 0.5 sy, 25.0 ni, 71.9 id, 0.7 wa, 0.0 hi, 1.4 si, 0.0 st

MiB Mem: 32114.1 total, 26888.8 free, 848.8 used, 4376.5 buff/cache

MiB Swap: 0.0 total, 0.0 free, 0.0 used. 29639.2 avail Mem

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM
7	user	38	18	931476	75376	33616	R	0.5	0.2
1	user	20	0	2500	588	520	S	0.0	0.0
6	user	38	18	2616	596	524	S	0.0	0.0
19	user	38	18	12184	6872	6040	S	0.0	0.0
406	user	38	18	7856	5928	3400	S	0.0	0.0
833	user	38	18	4308	520	452	T	0.0	0.0

### RESULT:

Thus, the basic Linux commands were executed and the output was verified successfully.



EX. NO: 6

## UTILIZATION OF GNU TOOL CHAINS OR EFFECTIVE SYSTEM PROGRAMMING

DATE:24/01/23

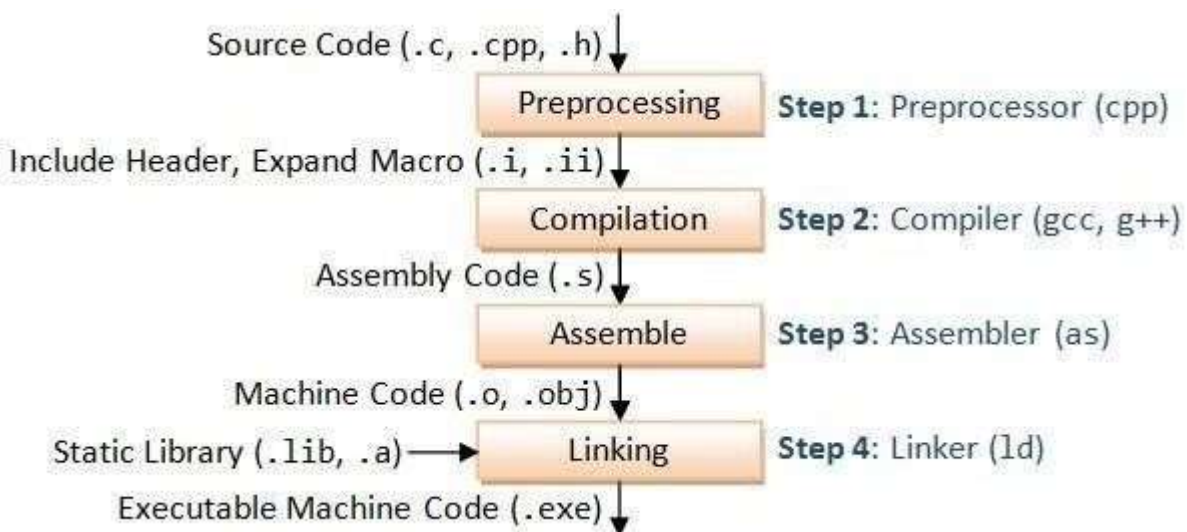
### AIM:

To compile the embedded c program using GNU toolchain.

### GCC:

GCC referred to the GNU C Compiler, but now refers to the GNU Compiler Collection. GCC is a collection of integrated compilers for the C, C++, Objective-C, Java, FORTRAN, and Ada programming. GCC is used to compile the source file stage by stage.

### GCC Compilation Process



GCC compiles a C/C++ program into executable in 4 steps as shown in the above diagram. Forexample,

"gcc -o hello.exe hello.c" is carried out as follows:

- Pre-processing: via the GNU C Preprocessor (cpp.exe), which includes the headers (#include) and expands the macros (#define).

```
cpp hello.c > hello.i
```

The resultant intermediate file "hello.i" contains the expanded source code.

- Compilation: The compiler compiles the pre-processed source code into assembly code for a specific processor.

```
gcc -S hello.i
```

The -S option specifies to produce assembly code, instead of object code. The resultant assembly file is "hello.s".

- Assembly: The assembler (as.exe) converts the assembly code into machine code in the object file "hello.o".

```
-o hello.o hello.s
```

- Linker: Finally, the linker (ld.exe) links the object code with the library code to produce an executable file "hello.exe".

```
-o hello.exe hello.o
```

```
gcc -o hello hello.o
```

If you have several files with source code ".c", you can compile them together, or you can do it in intermediate steps by creating first the object files ".o", and then linking them together to produce the final executable.

**Code:**

```
cpp hello.c > hello.i
```

```
gcc hello.c -o
```

```
hello.out & edit
```

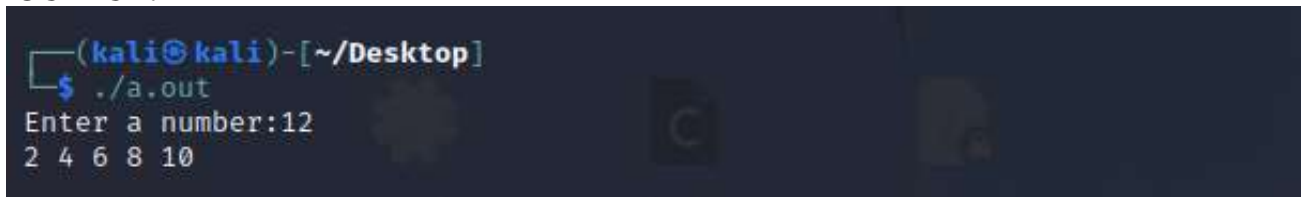
```
hello.i
```

```
./hello.out
```

### PROGRAM:

```
#include<stdio.h>
int main()
{
    int n,sum=0,m;
    printf("Enter a number:");
    scanf("%d",&n);
    for(int i=1;i<n;i++){
        if(i%2==0){
            printf("%d ",i);
        }
    }
    return 0;
}
```

### OUTPUT:



```
(kali@kali) - [~/Desktop]
$ ./a.out
Enter a number:12
2 4 6 8 10
```

### RESULT:

The C program to print hello world is compiled using GNU toolchain and output is verified successfully

EX. NO: 7	<b>BROWSING SOURCE PROGRAMS USING CSCOPE TOOL</b>
DATE:31/01/23	

### AIM:

To debug the project using CScope.

### PROCEDURE:

#### Step 1: Set Up the Environment

CSCOPE is a screen-oriented tool that can only be used on terminals listed in the TerminalInformation Utilities (terminfo) database. Be sure that set the TERM environment variable to your terminal type so that cscope can verify that it is listed in the terminfo database. Otherwise assign a value to TERM and export it to the shell as follows:

```
$ TERM=term_name; export TERM
```

In a C shell, type:

```
% setenv TERM term_name
```

#### Step 2: Invoke the cscope Program

By default, cscope builds a symbol cross-reference table for all the C, lex, and yacc source files in the current directory, and for any included header files in the current directory or the standard place. So, if all the source files for the program to be browsed are in the current directory, and if its header files are there or in the standard place, invoke cscope without arguments:

```
% cscope
```

To browse through selected source files, invoke cscope with the names of those files as arguments:

```
% cscope file1.c file2.c file3.h
```

For other ways to invoke cscope, cscope builds the symbol cross-reference table the first time it is used on the source files for the program to be browsed. By default, the table is stored in the file cscope.out in the current directory. On a subsequent invocation, cscope rebuilds the cross-reference only if a source file has been modified or the list of source files is different. When the cross-reference is rebuilt, the data for the unchanged files is copied from the old cross-reference, which makes rebuilding faster than the initial build, and reduces startup time for subsequent invocations.

### Step 3: Locate the Code

Table cscope Menu Manipulation Commands

S.No	Command	Description
1.	Tab	Move to the next input field.
2.	Return	Move to the next input field.
3.	^n	Move to the next input field.
4.	^p	Move to the previous input field.
5.	^y	Search with the last text typed.
6.	^b	Move to the previous input field and search pattern.
7.	^f	Move to the next input field and search pattern.
8.	^c	Toggle ignore/use letter case when searching. For example, a search for FILE matches file and File when ignoring the letter case.
9.	^r	Rebuild cross-reference.
10.	!	Start an interactive shell. Type ^d to return to cscope.
11.	^l	Redraw the screen.
12.	?	Display the list of commands.
13.	^d	Exit cscope.

If the first character of the text for which you are searching matches one of these commands, you can escape the command by entering a \ (backslash) before the character. Now move the cursor to the fifth menu item, Find this text string, enter the text out of storage, and press the Return key. cscope Function: Requesting a Search for a Text String:

cscope searches for the specified text, finds one line that contains it, and reports its finding.

cscope Function: Listing Lines Containing the Text String:

Text string: out of storage

File Line

1 alloc.c 63 (void) fprintf(stderr, "\n%s: out of storage\n", argv0); Find

this C symbol:

Find this global definition:

Find functions called by this function:

Find functions calling this function:

Find this text string:

Change this text string:

Find this egrep pattern:

Find this file:

Find files #including this file:

After cscope shows you the results of a successful search, you have several options. You may want to change one of the lines or examine the code surrounding it in the editor. Or, if cscope has found so many lines that a list of them does not fit on the screen at once, you may want to look at the next part of the list.

The following table shows the commands available after cscope has found the specified text:

Table Commands for Use After an Initial Search

S.No	Command	Description
1.	1 -9	Edit the file referenced by this line. The number you type corresponds to an item in the list of lines printed by cscope.
2.	Space	Display the next set of matching lines.
3.	+	Display the next set of matching lines.
4.	^v	Display the next set of matching lines.
5.	—	Display the previous set of matching lines.
6.	^e	Edit the displayed files in order.
7.	>	Append the list of lines being displayed to a file.
8.		Pipe all lines to a shell command.

Again, if the first character of the text for which you are searching matches one of these commands, you can escape the command by entering a backslash before the character.

The editor is invoked with the file alloc.c with the cursor at the beginning of line 63 of alloc.c.

cscope Function: Examining a Line of Code:

```
{      return(alloctest(realloc(p, (unsigned)
size)));
}
/* check for memory allocation failure
*/ static char * alloctest(p) char *p; {
if (p == NULL) {
(void) fprintf(stderr, "\n%s:  out of storage\n", argv0);
exit(1);
}
return(p);
```

}

"alloc.c" 67 lines, 1283 characters

## Output:

```
(kali㉿kali)-[~]
$ sudo apt install gedit
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docbook-xml fonts-dejavu gedit-common gir1.2-gtksource-4 gir1.2-peas-1.0 libamtk-5-0 libamtk-5-comm
  libjavascriptcoregtk-4.1-0 libpeas-1.0-0 libpeas-common libpython3.11 libpython3.11-minimal libpyth
  python3.11 python3.11-minimal sgml-base sgml-data xml-core yelp yelp-xsl
Suggested packages:
  docbook docbook-dsssl docbook-xsl docbook-defguide gedit-plugins glibc-doc libnss-nis libnss-nisplu
  perlsgml w3-recs opensp debhelper
Recommended packages:
  manpages-dev libc-devtools
The following NEW packages will be installed:
  docbook-xml fonts-dejavu gedit gedit-common gir1.2-gtksource-4 gir1.2-peas-1.0 libamtk-5-0 libamtk-
  libpython3.11 libpython3.11-minimal libpython3.11-stdlib libssl3 libtepl-6-2 libtepl-common libwebk
  yelp-xsl
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-l10n libc6 libc6-dev libc6-i386 locales
7 upgraded, 28 newly installed, 0 to remove and 1759 not upgraded.
Need to get 34.0 MB/59.1 MB of archives.
After this operation, 185 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

```
(kali㉿kali)-[~]
$ sudo apt install cscope
[sudo] password for kali:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  cscope-el
The following NEW packages will be installed:
  cscope
0 upgraded, 1 newly installed, 0 to remove and 1759 not upgraded.
Need to get 223 kB of archives.
After this operation, 1,260 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 cscope amd64 15.9-1 [223 kB]
Fetched 223 kB in 20s (11.0 kB/s)
Selecting previously unselected package cscope.
(Reading database ... 301968 files and directories currently installed.)
Preparing to unpack .../cscope_15.9-1_amd64.deb ...
Unpacking cscope (15.9-1) ...
Setting up cscope (15.9-1) ...
Processing triggers for kali-menu (2022.2.0) ...
Processing triggers for man-db (2.10.2-1) ...

(kali㉿kali)-[~]
$ cscope -v
```

```

File Actions Edit View Help
#include<stdio.h>
int main()
{
    int num,temp,fact=1;
    printf("Enter a number:");
    scanf("%d",&num);
    temp=num;
    while(num!=0)
    {
        fact*=num;
        num--;
    }
    printf("The factorial of %d is %d",temp,fact);
    return 0;
}

```

```

(kali@kali)-[~]
$ vim factorial.c

(kali@kali)-[~]
$ gcc factorial.c

(kali@kali)-[~]
$ ./a.out
Enter a number:7
The factorial of 7 is 5040

```

```

C symbol: a
File      Function Line
0 cprog.c main      3 int a = 12;
1 cprog.c main      5 printf("%d",a+b);

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:

```

```

Files #including this file: stdio.h
File      Line
0 cprog.c  1 #include <stdio.h>
1 factorial.c 1 #include <stdio.h>
2 odd_or_even.c 1 #include <stdio.h>
3 sum_of_nos.c 1 #include <stdio.h>
4 stdio.h   982 #include <bits/stdc++.h>

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:

```

```

File: odd_or_even
File
0 odd_or_even.c

Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:

```

## RESULT:

The C program to find factorial of a number and debugging is done using CScope tool.



EX. NO: 8

DATE:04/02/23

## CONSTRUCT CHARACTER ORIENTED DEVICE DRIVERS

### Aim:

To construct a simple character device driver program in Linux.

### Description:

- The device drivers are embedded software modules that contain the functionality to operate the individual hardware devices.
- The reason for the device driver software is to remove the need for the application to know how to control each piece of hardware.
- Each individual device driver would typically need to know only how to control its hardware device
- Character device drivers are used for driving sequential access devices. The amount of data accessed is not of fixed size.
- The character device drivers are accessed by the application using the standard calls such as open, read, write.
- The role of a driver is to provide mechanisms which allow normal users to access protected parts of its system, in particular ports, registers and memory addresses normally managed by the operating system.
- One of the good features of Linux is the ability to extend at runtime the set of the features offered by the kernel. Users can add or remove functionalities to the kernel while the system is running.
- These \programs" that can be added to the kernel at runtime are called \module" and built into individuals with .ko (Kernel object) extension.

The Linux kernel takes advantage of the possibility to write kernel drivers as modules which can be uploaded on request.

### Commands:

<i>Command</i>	<i>Description</i>
\$ uname -r	Returns a string naming the current system
\$ ls	To check object file created or not in the specified directory
\$ sudo dmesg	To see the message communicated by modules to the kernel
\$ sudo dmesg -C	To clear the communicated message
\$ sudo dmesg	To check message communication
\$ lsmod	List all the modules running in the systems

\$ sudo insmod simpleDriver.ko	(here simpleDriverf is user defined file.. ko kernel object file) It inserts the simpleDriver module in the list
-----------------------------------	---

\$ sudo rmmod simpleDriver.k	To remove kernel object (now the module is removed successfully check the command
---------------------------------	--

### **Code:**

#### ***hello.c***

```
#include <linux/module.h>
#include <linux/init.h>

/*META INFORMATION*/
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Raghav 4 GNU/Linux");
MODULE_DESCRIPTION("A hello world Linux kernal module");

// @brief This function is called, when the module is loaded into the kernel
static int __init hello_start(void)
{
    printk ("Hello, I'm here to help\n");
    return 0;
}

// @brief This function is called, when the module is removed into the kernel
static void __exit hello_end(void)
{
    printk("Goodbye, I hope I was helpful\n");
}

module_init(hello_start);
module_exit(hello_end);
```

#### ***Makefile:***

```
obj-m += hello.o
KVERSION = $(shell uname -r)
all:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
```

## Output:

```
ubuntu@ubuntu:~$ mkdir character_device_driver
ubuntu@ubuntu:~$ cd character_device_driver
ubuntu@ubuntu:~/character_device_driver$ gedit hello.c
ubuntu@ubuntu:~/character_device_driver$ gedit Makefile
ubuntu@ubuntu:~/character_device_driver$ make
make -C /lib/modules/5.15.0-58-generic/build M=/home/ubuntu/character_device_driver modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-58-generic'
  CC [M] /home/ubuntu/character_device_driver/hello.o
  MODPOST /home/ubuntu/character_device_driver/Module.symvers
  CC [M] /home/ubuntu/character_device_driver/hello.mod.o
  LD [M] /home/ubuntu/character_device_driver/hello.ko
  BTF [M] /home/ubuntu/character_device_driver/hello.ko
Skipping BTF generation for /home/ubuntu/character_device_driver/hello.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-58-generic'
ubuntu@ubuntu:~/character_device_driver$ ls
hello.c  hello.mod  hello.mod.o  Makefile  Module.symvers
hello.ko  hello.mod.c  hello.o  modules.order
ubuntu@ubuntu:~/character_device_driver$ sudo insmod hello.ko
[sudo] password for ubuntu:
```

```
Open  ▾  [F]  hello.c
~/character_device_driver

1 #include <linux/module.h>
2 #include <linux/init.h>
3
4 /*META INFORMATION*/
5 MODULE_LICENSE("GPL");
6 MODULE_AUTHOR("Raghav 4 GNU/Linux");
7 MODULE_DESCRIPTION("A hello-world Linux kernel module");
8
9 // @brief This function is called, when the module is loaded into the kernel
10 static int __init hello_start(void)
11 {
12     printk("Hello, I'm here to help\n");
13     return 0;
14 }
15
16 // @brief This function is called, when the module is removed into the kernel
17 static void __exit hello_end(void)
18 {
19     printk("Goodbye, I hope I was helpful\n");
20 }
21
22 module_init(hello_start);
23 module_exit(hello_end);
24
```

```
Open  ▾  [F]  Makefile
~/character_device_driver

1 obj-m += hello.o
2 KVERSION = $(shell uname -r)
3 all:
4     make -C /lib/modules/$(KVERSION)/build M=$(PWD) modules
5 clean:
6     make -C /lib/modules/$(KVERSION)/build M=$(PWD) clean
7
```

```
ubuntu@ubuntu:~/character_device_driver$ lsmod
Module                  Size  Used by
hello                   16384  0
vsock_loopback          16384  0
vmw_vsock_virtio_transport_common 40960  1 vsock_loopback
vmw_vsock_vmci_transport 32768  2
vsock                   45056  7 vmw_vsock_virtio_transport_common,vsock_loopback,vmw_vsock_vmci_transport
nls_iso8859_1            16384  1
snd_ens1371             32768  2
snd_ac97_codec          155648  1 snd_ens1371
binfmt_misc             24576  1
gameport                24576  1 snd_ens1371
ac97_bus                16384  1 snd_ac97_codec
intel_rapl_msr          20480  0
intel_rapl_common       40960  1 intel_rapl_msr
snd_pcm                 135168  2 snd_ac97_codec,snd_ens1371
snd_seq_midi            20480  0
crct10dif_pclmul        16384  1
snd_seq_midi_event      16384  1 snd_seq_midi
ghash_clmulni_intel     16384  0
snd_rawmidi             49152  2 snd_seq_midi,snd_ens1371
vmw_balloon            24576  0
aesni_intel            376832  0
crypto_smd              16384  1 aesni_intel
cryptd                  24576  2 crypto_smd,ghash_clmulni_intel
snd_seq                 77824  2 snd_seq_midi,snd_seq_midi_event
snd_seq_device          16384  3 snd_seq,snd_seq_midi,snd_rawmidi
snd_timer              40960  2 snd_seq,snd_pcm
i2cdev                  32768  0
```

```
ubuntu@ubuntu:~/character_device_driver$ lsmod | grep hello
hello                   16384  0
```

```

ubuntu@ubuntu:~/character_device_driver$ dmesg | tail
[ 12.935300] rfkill: input handler disabled
[ 83.441481] rfkill: input handler enabled
[ 90.273891] rfkill: input handler disabled
[ 878.621137] perf: interrupt took too long (2768 > 2500), lowering kernel.perf_event_max_sample_rate to 72250
[ 1014.491648] hello: loading out-of-tree module taints kernel.
[ 1014.491683] hello: module verification failed: signature and/or required key missing - tainting kernel
[ 1014.492388] Hello, I'm here to help
[ 1100.525380] Goodbye, I hope I was helpful
[ 1230.154669] perf: interrupt took too long (3519 > 3460), lowering kernel.perf_event_max_sample_rate to 56750
[ 1433.640951] Hello, I'm here to help
ubuntu@ubuntu:~/character_device_driver$ sudo rmmod hello
ubuntu@ubuntu:~/character_device_driver$ dmesg | tail
[ 83.441481] rfkill: input handler enabled
[ 90.273891] rfkill: input handler disabled
[ 878.621137] perf: interrupt took too long (2768 > 2500), lowering kernel.perf_event_max_sample_rate to 72250
[ 1014.491648] hello: loading out-of-tree module taints kernel.
[ 1014.491683] hello: module verification failed: signature and/or required key missing - tainting kernel
[ 1014.492388] Hello, I'm here to help
[ 1100.525380] Goodbye, I hope I was helpful
[ 1230.154669] perf: interrupt took too long (3519 > 3460), lowering kernel.perf_event_max_sample_rate to 56750
[ 1433.640951] Hello, I'm here to help
[ 1539.310597] Goodbye, I hope I was helpful

```

```

ubuntu@ubuntu:~/character_device_driver$ modinfo hello.ko
filename:          /home/ubuntu/character_device_driver/hello.ko
description:       A hello world Linux kernel module
author:           Raghav 4 GNU/Linux
license:          GPL
srcversion:       7CC42D0B45E4422A5624400
depends:
retpoline:        Y
name:             hello
vermagic:         5.15.0-58-generic SMP mod unload modversions

```

## Result:

The C program is written to create Character Device Driver program and output is verified successfully.

EX.NO: 9 A	<b>IMPLEMENTATION OF TASK MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II</b>
DATE:07/02/23	

**Aim:**

To develop a C program for creating tasks using FreeRTOS APIs.

**Software Requirement:**

- Ubuntu Linux distros
- Text editors like gedit, vi in linux with gcc

**Description:**

- In FreeRTOS, an application can consist of many tasks. If the processor running the application contains a single core, then only one task can be executing at any given time. This implies that a task can exist in one of two states, Running and Not Running.
- When a task is in the Running state the processor is executing the task's code. When a task is in the Not Running state, the task is dormant, its status having been saved ready for it to resume execution the next time the scheduler decides it should enter the Running state.
- The FreeRTOS scheduler is the only entity that can switch a task in and out.

**Creating Tasks: The xTaskCreate() API Function**

- Tasks are created using the FreeRTOS xTaskCreate() API function.

```

BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        uint16_t usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );

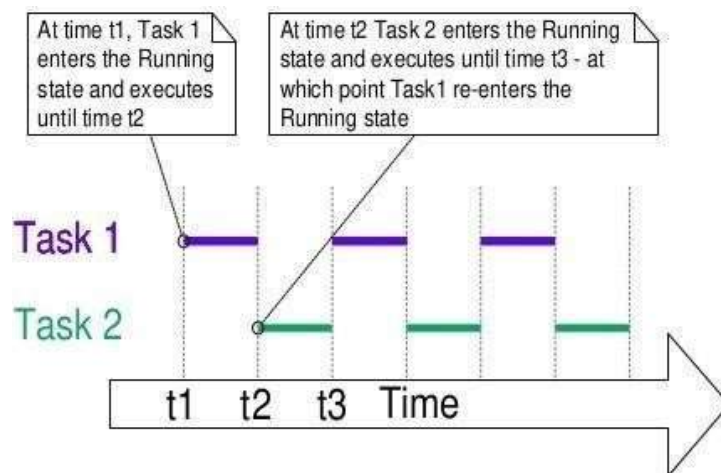
```

- pvTaskCode parameter is simply a pointer to the function that implements the task (in effect, just the name of the function).
- pcName A descriptive name for the task. this is not used by FreeRTOS in any way. It is included purely as a debugging aid. Identifying a task by a human readable name is much simpler than attempting to identify it by its handle.
- usStackDepth Each task has its own unique stack that is allocated by the kernel to the task when the task is created. The usStackDepth value tells the kernel how large to make the stack.
- pvParameters Task functions accept a parameter of type pointer to void ( void\* ). The value assigned to pvParameters is the value passed into the task.
- uxPriority Defines the priority at which the task will execute. Priorities can be assigned from 0, which is the lowest priority, to (configMAX\_PRIORITIES – 1), which is the highest priority.
- pxCreatedTask This can be used to pass out a handle to the task being created. This handle can then be used to reference the task in API calls that, for example, change the task priority

or delete the task. If your application has no use for the task handle, then `pxCreatedTask` can be set to `NULL`.

### Returned value

- `pdPASS` This indicates that the task has been created successfully.
- `pdFAIL` This indicates that the task has not been created because there is insufficient heap memory available for FreeRTOS to allocate enough RAM to hold the task data structures and stack.
- In the below program, two tasks (Task 1 & Task 2) are created as follows:



### Procedure:

- Install the dependencies for Ubuntu  
`sudo apt-get install libcs6-dev-i386`
- Navigate to the C source code  
`$cd Project/main.c`
- Compile the Makefile using make command  
`$make`

### Program:

```
// Task Creation
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<FreeRTOS.h>
#include<task.h>
void vTask1(void*);
void vTask2(void*);
void vApplicationIdleHook(void);
int main(void)
{
    xTaskCreate(vTask1,"Task1",1000,NULL,1,NULL);
    xTaskCreate(vTask2,"Task2",1000,NULL,1,NULL);
    vTaskStartScheduler();
    return 0;
}
```

```
void vAssertCalled( unsigned long ulLine, const char * const pcFileName )
{
    taskENTER_CRITICAL();
    {
        printf("[ASSERT] %s:%lu\n", pcFileName, ulLine);
        flush(stdout);
    }
    taskEXIT_CRITICAL();
    exit(-1);
}

void vTask1(void* parameter)
{
    while(1)
    {
        printf("Task1\n");
        sleep(500);
    }
}

void vTask2(void* parameter)
{
    while(1)
    {
        printf("Task2\n");
        sleep(500);
    }
}

void vApplicationIdleHook(void)
{
    //printf("Idle\r\n");
}
```

**Output:**

[illegible]

**Result:**

Thus, task creation using FreeRTOS API functions is executed and the output is verified successfully.



EX.NO: 9 B	<b>IMPLEMENTATION OF INTERUPPT MANAGEMENT IN REAL-TIME OPERATING SYSTEMS (RTOS) USING MICROC/OS-II</b>
DATE:14/02/23	

### **Aim:**

To develop a C program for scheduling tasks based on “Round Robin algorithm” using FreeRTOS APIs.

### **Software Requirement:**

- Ubuntu Linux distros
- Text editors like gedit, vi in linux with gcc

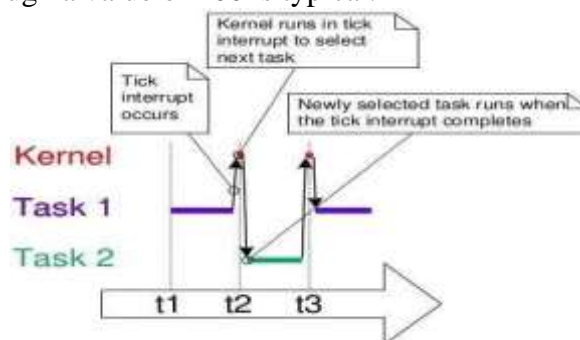
### **Description:**

#### **Task Priorities:**

- The priority can be changed after the scheduler has been started by using the `vTaskPrioritySet()` API function.
- Priorities are defined in `configMAX_PRIORITIES` compile time configuration constant within `FreeRTOSConfig.h`.
- Therefore, the range of available priorities is 0 to (`configMAX_PRIORITIES - 1`).
- The FreeRTOS scheduler will always ensure that the highest priority task that is able to run is the task selected to enter the Running state. Where more than one task of the same priority is able to run, the scheduler will transition each task into and out of the Running state, in turn.

#### **Time Measurement and the Tick Interrupt:**

- Scheduling Algorithms, describes an optional feature called ‘time slicing’ to be able to select the next task to run, the scheduler itself must execute at the end of each time slice 1.
- A periodic interrupt, called the ‘tick interrupt’, is used for this purpose.
- `configTICK_RATE_HZ` compile time configuration constant within `FreeRTOSConfig.h`.
- `configTICK_RATE_HZ` is set to 100 (Hz), then the time slice will be 10 milliseconds. The time between two tick interrupts is called the ‘tick period’. One time slice equals one tick period.
- The optimal value for `configTICK_RATE_HZ` is dependent on the application being developed, although a value of 100 is typical.

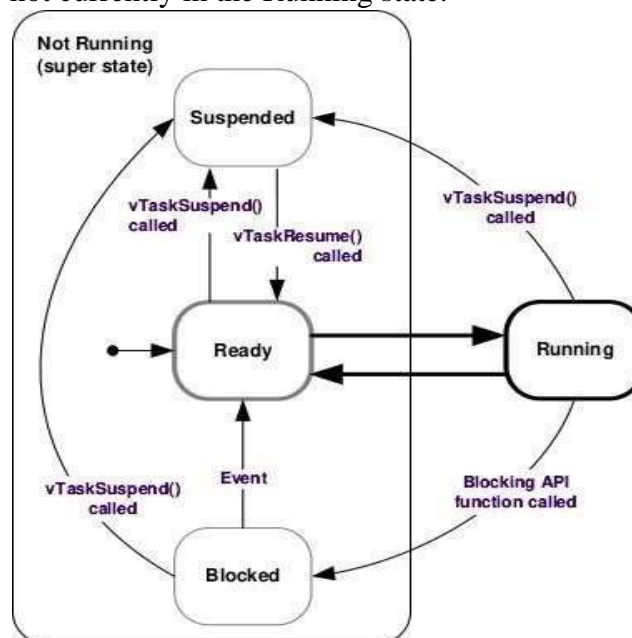


- FreeRTOS API calls always specify time in multiples of tick periods, which are often referred to simply as ‘ticks’. The `pdMS_TO_TICKS()` macro converts a time specified in milliseconds into a time specified in ticks.

*TickType\_t xTimeInTicks = pdMS\_TO\_TICKS( 200 );*

### Expanding the ‘Not Running’ State:

- To make the tasks useful they must be re-written to be event-driven. An event-driven task has work (processing) to perform only after the occurrence of the event that triggers it, and is not able to enter the Running state before that event has occurred.
- A task that is waiting for an event is said to be in the ‘Blocked’ state, which is a sub-state of the Not Running state.
- Temporal (time-related) events—the event being either a delay period expiring, or an absolute time being reached. For example, a task may enter the Blocked state to wait for 10 milliseconds to pass.
- Synchronization events—where the events originate from another task or interrupt. For example, a task may enter the Blocked state to wait for data to arrive on a queue. Synchronization events cover a broad range of event types.
- The Suspended State ‘Suspended’ is also a sub-state of Not Running. Tasks in the Suspended state are not available to the scheduler. The only way into the Suspended state is through a call to the `vTaskSuspend()` API function, the only way out being through a call to the `vTaskResume()` or `xTaskResumeFromISR()` API functions.
- The Ready State Tasks that are in the Not Running state but are not Blocked or Suspended are said to be in the Ready state. They are able to run, and therefore ‘ready’ to run, but are not currently in the Running state.



- `vTaskDelay()` places the calling task into the Blocked state for a fixed number of tick interrupts.

*void vTaskDelay( TickType\_t xTicksToDelay );*

- `vTaskDelay( pdMS_TO_TICKS( 100 ) )` will result in the calling task remaining in the Blocked state for 100 milliseconds.

### The vTaskDelayUntil() API Function

- The parameters to vTaskDelayUntil() specify, instead, the exact tick count value at which the calling task should be moved from the Blocked state into the Ready state.
- vTaskDelayUntil() is the API function that should be used when a fixed execution period is required (where you want your task to execute periodically with a fixed frequency), as the time at which the calling task is unblocked is absolute, rather than relative to when the function was called (as is the case with vTaskDelay()).
- void vTaskDelayUntil( TickType\_t \* pxPreviousWakeTime, TickType\_t xTimeIncrement );
- pxPreviousWakeTime : This parameter is named on the assumption that vTaskDelayUntil() is being used to implement a task that executes periodically and with a fixed frequency. In this case, pxPreviousWakeTime holds the time at which the task last left the Blocked state (was 'woken' up). This time is used as a reference point to calculate the time at which the task should next leave the Blocked state.
- xTimeIncrement This parameter is also named on the assumption that vTaskDelayUntil() is being used to implement a task that executes periodically and with a fixed frequency—the frequency being set by the xTimeIncrement value.
- The xLastWakeTime variable needs to be initialized with the current tick count. Note that this is the only time the variable is explicitly written to. After this xLastWakeTime is managed automatically by the vTaskDelayUntil() API function.

### The Idle Task and the Idle Task Hook

There must always be at least one task that can enter the Running state. To ensure this is the case, an Idle task is automatically created by the scheduler when **vTaskStartScheduler()** is called.

- The idle task has the lowest possible priority (priority zero), to ensure it never prevents a higher priority application task from entering the Running state.

### Idle Task Hook Functions

- To add application specific functionality directly into the idle task through the use of an idle hook (or idle callback) function—a function that is called automatically by the idle task once per iteration of the idle task loop.
- Placing the processor into a low power mode.
- An Idle task hook function must never attempt to block or suspend.
- Idle task is responsible for cleaning up kernel resources after a task has been deleted. If the idle task remains permanently in the Idle hook function, then this clean-up cannot occur.

**void vApplicationIdleHook( void );**

### Procedure:

- Install the dependencies for Ubuntu  

```
sudo apt-get install libcs6-dev-i386
```
- Navigate to the C source code  

```
$cd Project/main.c
```
- Compile the Makefile using make command  

```
$make
```

**Program:**

```
// Task Scheduling using Round Robin algorithm
#include <stdio.h>
#include <stdlib.h>
#include <FreeRTOS.h>
#include <task.h>
#include <timers.h>
#define TASKSCHEDULER
#ifdef TASKSCHEDULER
void vTask1(void*);
void vTask2(void*);
void vTask3(void*);
void vTask4(void*);
#endif
void vApplicationIdleHook(void);
int main(void)
{
    #ifndef TASKSCHEDULER
    xTaskCreate( vTask1, "Task 1", 1000, NULL, 1, NULL );
    xTaskCreate( vTask2, "Task 2", 1000, NULL, 1, NULL );
    xTaskCreate( vTask3, "Task 3", 1000, NULL, 1, NULL );
    xTaskCreate( vTask4, "Task 4", 1000, NULL, 1, NULL );
    #endif
    vTaskStartScheduler();
    return 0;
}

void vAssertCalled( unsigned long ulLine, const char * const pcFileName )
{
    taskENTER_CRITICAL();
    {
        printf("[ASSERT] %s:%lu\n", pcFileName, ulLine);
        flush(stdout);
    }
    taskEXIT_CRITICAL();
    exit(-1);
}

#ifdef TASKSCHEDULER
void vTask1(void* parameter)
{
    while(1)
    {
        printf("Task 1\n");
        vTaskDelay(pdMS_TO_TICKS(250));
    }
}
```

```
void vTask2(void* parameter)
{
    while(1)
    {
        printf("Task 2\n");
        vTaskDelay(pdMS_TO_TICKS(250));
    }
}

void vTask3(void* parameter)
{
    TickType_t xLastWaketime = xTaskGetTickCount(); while(1)
    {
        printf("Task 3 with 250ms\n");
        vTaskDelayUntil(&xLastWaketime, pdMS_TO_TICKS(250));
    }
}

void vTask4(void* parameter)
{
    TickType_t xLastWaketime = xTaskGetTickCount();
    while(1)
    {
        printf("Task 4 with 500ms\n");
        vTaskDelayUntil(&xLastWaketime, pdMS_TO_TICKS(500));
    }
}

#endif

void vApplicationIdleHook(void)
{
    // printf("Idle\r\n");
}
```

## Output:

```
BUILD COMPLETE: FreeRTOS-Sim
-----
est@est:~/1802258/pmvankerFreeRTOS$ ./FreeRTOS-Sim
Running as PID: 14919
Timer Resolution for Run TimeStats is 100 ticks per second.
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
Task 3 with 250ms
Task 4 with 500ms
Task 1
Task 2
Task 3 with 250ms
Task 1
Task 2
```

## Result:

Thus, tasks were created and scheduled based on “Round Robin algorithm” using FreeRTOS APIs and the output is verified successfully.

EX.NO: 10 A	<b>DEVELOPMENT OF BLUETOOTH INTERFACING USING REAL TIME APPLICATION MSP430 LAUNCHPAD</b>
DATE: 21/02/23	

**Aim:**

To write a sketch program to connect the Bluetooth Module with MSP430G2553 to control a LED.

**Apparatus Required:**

- MSP430G2553 Launchpad
- Energia IDE
- HC-05 Bluetooth module.

**Procedure:**

- Attach the MSP430G2553 board with the system.
- Attach the Bluetooth Module with MSP430G2553 board.
- Double click on Energia IDE on the desktop.
- Select the board type as MSP430G2553 Launchpad from Tool.
- Create a new program on Energia IDE and save it.
- Compile the program and upload it to the MSP430G2553 Launchpad board.
- Run the program and verify the output by controlling the LED using an Android application on mobile.

**Program:**

```
#define LED RED_LED
void setup()
{
    Serial.begin(9600);
    pinMode(2, OUTPUT);
}
void loop()
{
    if (Serial.available())
    {
        char data_received;
        data_received = Serial.read();
        if (data_received == '1')
        {
            digitalWrite(LED, HIGH);
            Serial.write("LED turned ON\n");
        }
    }
}
```

```
if (data_received == '2')  
{  
    digitalWrite(LED, LOW);  
    Serial.write("LED turned OFF\n");  
}  
}  
}
```

### Output:



### Result:

Thus, the sketch program to connect the Bluetooth Module with MSP430G2553 to control a led was implemented successfully.



EX.NO: 10 B	<b>DEVELOPMENT OF ESP8266 INTERFACING (WIFI) USING MSP430 LAUNCHPAD</b>
DATE: 28/02/23	

**Aim:**

To write a sketch program to connect the ESP8266 WiFi Module with MSP430G2553 to send a data to browser.

**Apparatus Required:**

- MSP430G2553 Launchpad
- Energia IDE
- ESP8266 WiFi module.

**Procedure:**

- Attach the MSP430G2553 board with the system.
- Attach the WiFi Module with MSP430G2553 board.
- Double click on Energia IDE on the desktop.
- Select the board type as MSP430G2553 Launchpad from Tool.
- Create a new program on Energia IDE and save it.
- Compile the program and upload it to the MSP430G2553 Launchpad board.
- Run the program and verify the output by sending data to browser.

**Program:**

```
#define SSID "RAGHAV"
#define PASS "12345678"
#define DST_IP "things.ubidots.com"
#define idvariable "569fc4ba76254229c49896a6"
int len;

void setup()
{
    // Open serial communications and wait for port to open:
    char cmd[254];
    Serial.begin(9600);
    Serial.setTimeout(5000);
    //test if the module is ready
    Serial.println("AT+RST");
    delay(1000);
    if (Serial.find("ready"))
    {
        Serial.println("Module is ready");
    }
}
```

```

else
{
    Serial.println("Module have no response."); while (1);
}
delay (1000);

//connect to the wifi
boolean connected = false;
for (int i = 0; i < 5; i++)
{
    if (connectWiFi())
    {
        connected = true;
        break;
    }
}

if (!connected) {
    while (1);
}

delay(5000);
Serial.println("AT+CIPMUX=0");
}

void loop()
{
    int value = analogRead(A0); //you can change ir to another pin
    int num=0;
    String var = "{" + "value\":" + String(value) + "}";
    num = var.length();
    String cmd = "AT+CIPSTART=\"TCP\",";
    cmd += DST_IP;
    cmd += "\",80";
    Serial.println(cmd);
    if (Serial.find("Error"))
        return;
    len=strlen ("POST /api/v1.6/datasources/");
    len=len+strlen (idvariable);
    len=len+strlen ("/values HTTP/1.1\nContent-Type: application/json\nContent-Length:");
    char numlength[4]; // this will hold the length of num which is the length of the JSON
    element
    sprintf(numlength, "%d", num); // saw this clever code off the net; works yay len=len+strlen
    (numlength);
    len=len + num; //fixed length of the string that will print as Content-Length: in the POST

```

```

len=len+strlen ("\nX-Auth-Token: ");
len=len+strlen (token);
len=len+strlen ("\nHost: things.ubidots.com\n\n"); len=len+strlen ("\n\n");
Serial.print("AT+CIPSEND=");
Serial.println (len); //length of the entire data POST for the CIPSEND command of ESP2866
//Serial.println(cmd.length()); if (Serial.find(">"))
{
    //Serial.print(">");
}
else
{
    Serial.println("AT+CIPCLOSE"); delay(1000);
    return;
}

Serial.print ("POST /api/v1.6/variables/"); Serial.print (idvariable);
Serial.print ("/values HTTP/1.1\nContent-Type: application/json\nContent-Length: ");
Serial.print (num);
Serial.print ("\nX-Auth-Token: "); Serial.print (token);
Serial.print ("\nHost: things.ubidots.com\n\n"); Serial.print (var);
Serial.println ("\n\n"); delay(9000);
//Serial.find("+IPD"); clear the input buffer after the web site responds to the POST while
(Serial.available())
{
    char c = Serial.read();
}
delay(1000);
}

boolean connectWiFi()
{
    Serial.println("AT+CWMODE=1"); String cmd = "AT+CWJAP="; cmd += SSID;
    cmd += "\",\""; cmd += PASS; cmd += "\""; Serial.println(cmd); delay(2000);
    if (Serial.find("OK")){

```

```
        return true;
    }
    else{
        return false;
    }
```

### **Output:**



### **Result:**

Thus, the sketch program to connect a ESP8266 Wifi Module with MSP430G2553 to send a data to browser was implemented successfully.

EX.NO: 11	<b>DEVELOPING OF LED BLINKING EMBEDDED SYSTEM</b> <b>APPLICATION USING TI CC3200 LAUNCHPAD</b>
DATE:14/03/23	

**Aim:**

To write a CC3200 sketch for blinking (ON/OFF) of inbuilt LED using CC3200.

**Apparatus Required:**

- Energia IDE
- CC3200 board
- LED
- Bread board
- 220 ohm resistor
- Jumper wires

**Procedure:**

- Attach the CC3200 board with the system.
- Interface LED circuit with CC3200 board.
- Double click on Energia on the desktop.
- Select the board type as CC3200 from Tools-Board and also select COM port number from the PORT option
- Create a new program in the Energia IDE software and save it.
- Compile the program and upload it to the CC3200 board.
- Run the program and verify the output.

**Code:**

***For Single LED Bulb:***

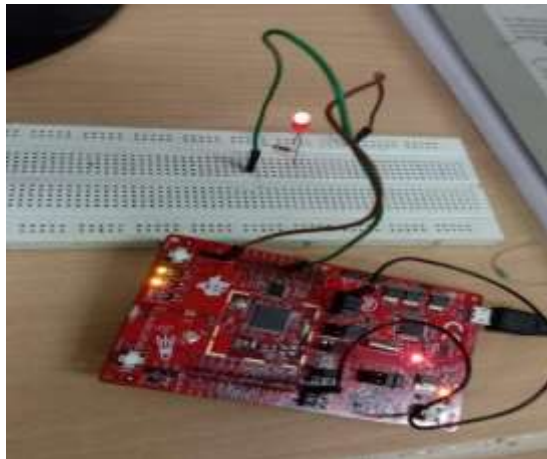
```
#define LED 5
void setup()
{
    pinMode(LED,OUTPUT);
}
void loop()
{
    digitalWrite(LED, HIGH);
    delay(1000);
    digitalWrite(LED,LOW);
    delay(1000);
}
```

***For Multiple LED Bulb:***

```

#define RLED 9
#define GLED 10
#define YLED 29 void
setup()
{
    pinMode(RLED,OUTPUT);
    pinMode(GLED,OUTPUT);
    pinMode(YLED,OUTPUT);
}
void loop()
{
    digitalWrite(RLED, HIGH);
    digitalWrite(GLED, HIGH);
    digitalWrite(YLED, HIGH); delay(1000);
    digitalWrite(RLED,LOW);
    digitalWrite(GLED,LOW);
    digitalWrite(YLED,LOW); delay(1000);
}

```

**Output:****Result:**

Thus, the Energia sketch to ON/OFF of built-in LEDs was executed successfully.

EX. NO: 12	DEVELOPMENT OF IOT APPLICATION WITH SENSORS USING TI CC3200 LAUNCHPAD
DATE:21/03/23	

**Aim:**

To write a program in Energia to check whether any live object traces are present by using Sensor using CC3200.

**Apparatus Required:**

- Energia IDE
- CC3200 Board
- LED
- PIR Sensor
- Breadboard

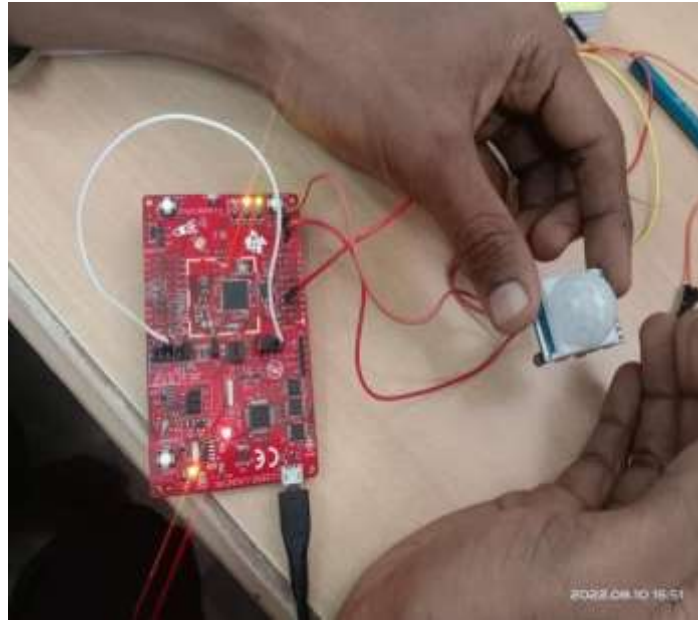
**Procedure:**

- Attach the CC3200 board to the system.
- Connect the PIR sensor with the digital pin of the CC3200 board.
- Double-click on Energia on the desktop.
- Select the board type as CC3200 from Tools-Board and also select the COM port number from the PORT option.
- Create a new program in the Energia software and save it.
- Compile the program and upload it to the CC3200 board.
- Run the program and verify the output.

**Program:**

```
int pir=4;
int val = LOW;
void setup()
{
    pinMode(pir, INPUT);
    Serial.begin(9600);
}
void loop() {
    val = digitalRead(pir);
    if (val == HIGH)
    {
        Serial.println("Motion Detected");
    }
    else
    {
        Serial.println("Motion NOT Detected");
    }
}
```

### **Output:**



### **Result:**

Thus, the Energia sketch to interface the Sensor with CC3200 was executed and the output was verified successfully.