ExtendedC

Version 1.0

Manuel

By Malith Perera

Introduction

Extended C allows you to do every thing what C does and extend C functionalities to create easy variables, arrays, lists, queues, stacks. It also help you to manage dynamic memory if you are interested in. That's not all it supports develop your application with objects in mind.

Here you need basic knowledge of C programming concepts to move further. If you unfamiliar with C language concepts please download `C Basics' from the link at the end of this document. It may take no more than an hour if you know the basic programming language concepts.

What you have to do here is follow the instructions in the document and study the code given in the examples. Then compile and run. Inspect the code carefully till you get understand. Important parts in the code are highlighted. I'm sure you will grab the essence. It's simple and easy. Same thing is repeating again and again.

Next add some codes to your file or create new files as you like. Write your own code what you get from the example. Then save and compile. Play with your code until you can do something alone.

Here we use GCC compiler in Linux terminal with vim editor. It's free. So you can give it a try. Else you can use MingW compiler and VSCode as editor in Windows or GCC and XCode in Mac OS. Please search internet how to install, edit, compile and run C programs in your environment.

Play with your code and enjoy.

Happy Coding!

Support EC

This simple document is written in LibreOffice Writer which is also a free and open source software. This is a free document you can use it, change it, enhance it as you like.

Because use of minimum words you can translate this to your native language. It will be a great help for other who speaks your language. Anyone who translate, fix or design this document more colorful do not forget to add your name to editors page below. Please keep this document up to date and as simple as possible.

If you find any error in this document or even the source code of EC you can fix it and save it on EC directory. While updating EC it will upload the fixes or changes automatically to to the world.

Thank you!

Community Support

Please join with our community and help others and ask question what you get.

Download EC

You can use EC with or without installing it into your system. Though It's not necessary to install, you just need to download and keep it beside or within your application. Also you can download binary files for Unix/Linux, Windows and Mac OS from the download link given below.

www.eclab.org/download

Written with : LibreOffice Writter

Font Styles : URW Gothic
`Liberation Mono`

Editors Page

Edited by
Malith Perera 2024

Fixed by
Malith Perera 2024

Designed by
Malith Perera 2024

Content

Title **Page**

How to start

Directly from the source

You can directly start with the downloaded source without installing. First move to the EC directory. Then run ec according to your operating system as below.

Unix/Linux

```
$ cd EC
$ ./ec
```

Windows

```
c:\\> cd EC
c:\\> .\ec.exe
```

Mac OS

```
$ cd EC
$ ./ec.mac
```

After given above command in the terminal or shell you should see, Welcome message and ask the user name.

```
$ ./ec

EC 0.0.0 on Linux 6.5.0-28-generic x86_64

--------------------------------------------
| Welcome To the EC World! |
--------------------------------------------

Please enter user name:
```

If you do not see above you should compile and build it from the source as below.

Compiling the source

To compile from the source you must have installed compiler as gcc, clang or any other compiler Then run the shell script or batch script as below.

For Unix/Linux

```
$ ./.ec/ec_make.sh
```

For Windows

```
c:\\> .\ec.exe\ec_make.batch
```

For Mac OS

```
$ ./.ec/ec_make_mac.sh
```

Installing EC

Before install you should put EC into a permanent directory where your projects are going to be created. After installation do not change the place or rename the parent directory. If you change these do not forget to reinstall it.

Directory Tree

```
Projects directory
   |
   |- EC
   |- MyApp1
   |- MyApp2
```

To install first move to EC directory from your terminal or shell program and follow the commands below.

For Unix/Linux

```
$ ./ec install
```

For Windows

```
c:\\> .\ec.exe install
```

For Mac OS

```
$ ./ec.mac install
```

Note:

If you working from the source move to EC directory or your application directory and use ./ec to run EC. Here `./' mean in the ec is in the current working directory

```
$ ./ec
```

If you installed EC you can use ec without `./'.

```
$ ec
```

Welcome to EC

If you already install EC you can open the command line terminal and check the first run.

```
$ ec

-----------------------------------
| Welcome To be EC! |
-----------------------------------

 Please enter user name:
```

Give an user name and follow the instructions to signin to EC as a user.

Creating your first application

Open the terminal or shell and enter the following command.

```
$ ec app MyApp
```

Your first application `MyApp´ will be created outside the EC directory when you working from the source or in the working directory if you have installed EC to your system.

Directory Tree

```
Projects directory
   |
   |- EC
   |- MyApp
   |- MyApp2
```

Compile and Run

Move to the MyApp directory and run ec as below.

```
$ ec
```

That's all. It will compile and run your application and create your application executable `MyApp´ in MyApp directory.

Run

You can run MyApp without compiling as below.

```
$ ./MyApp
```

If you know C language now you can change the source code as you like and then compile it and run it.

EC Todo

Now you know how to compile and run your source easily. But most of the time it's an easy way to develop by tracking your development path by adding and fulfilling tasks one by one. These can be done by your self or with your development team too. You can add new tasks as additions or bug fixes to your project as below

Todo task options

1. additions (-a)
2. bugfixes (-b)

Todo urgency options

1. Urgent (-u)
2. Essential (-e)
3. Required (-r)
4. Optional (-o)

Display todo list

```
$ ec todo
```

Adding a task

```
$ ec todo  `Write a function to print pationts list'
```

Adding a task with options

```
$ ec todo -ae "Create function for add and remove names"
```

Here -ae mean essential addition.

Examples:
-bu -> urgent bugfix
-ao -> optional addition

Removing a task

```
$ ec todo -r task number
```

The task number will be displayed in front of every task while displaying Todo list.

Assign to a task

```
$ ec todo -s (task number)
```

Submit a completed task

```
$ ec todo -s (task number)
```

Be able to submit only who assigned to it. Otherwise ask to assign. It'll be add to EC Workload.

Todo help

```
$ ec todo -h
```

EC Workload

Display what has done, did by whom, when, duration and also a summery of the project.

Display workload summery

```
$ ec workload
```

Display workload user by user

```
$ ec workload -u
```

Workload help

```
$ ec workload -h
```

Hello world C

```
#include <stdio.h>


int main(int argc, char *argv[])
{
     printf(“Hello world!”);

     return 0;
}
```

Introduction to C structures

C structures allows to combine various data types into a one place.

Structure syntax

```
struct Student {
    char *name;
    int   age;
};
```

Structure with typedef

```
typedef struct Student {
    char *name;
    int   age;
} Student;
```

Here define a type called Student. We usually use structure syntax with typedef as above in EC.

Note: If you want to know more about C structures you can learn `C Basics' from the link below.

http://www.eclab.org/doc/c_basics

EC variable

EC allows you to create dynamic variable easily as below.

Syntax overview

```
#include "ec.h"



int main(int argc, char *argv[])
{
    char *st1 = Student_New();

    *st1 = (Student) {"Arnold", 63};

    printf("%s is %d years old\n", st->name, st->age);

    return 0;

}
```

EC Array

Syntax overview

```
#include "ec.h"

typedef struct Student {
    char *name;
    int   age;
} Student;


EC(Student)

int main(int argc, char *argv[])
{
     StudentArray *student_array = Student_Array(2);

     student_array->var[0] = (Student) {"Arnold", 12};
     student_array->var[1] = (Student) {"Silvester", 10};

     Student *st;
     for_array(student_array, st)
     {
          printf("%s is %d years old\n", st->name, st->age);
     }

     return 0;
}
```

EC List

EC List lists variables order as you wish. Good for random accesses and deletions of variables.

Syntax overview

```
#include "ec.h"

typedef struct Student {
    char *name;
    int   age;
} Student;


EC(Student)


int main(int argc, char *argv[])
{
    StudentList *student_list = Student_List();

    Student *student1 = Student_New();
    Student *student2 = Student_New();

    *student1 = (Student) {"Arnold", 12};
    *student2 = (Student) {"Silvester", 10};

    Student_Append(student_list, student1);
    Student_Append(student_list, student2);

    Student *st;
    for_list(student_list, st) {
        printf("%s age is %d\n", st->name, st->age);

     }

     return 0;
}
```

EC Queue

EC Queue queues variables first to last. Only serves the variables as order in the queue in first serves first nature. Good for sequential access rather than random accesses.

Syntax overview

```
#include "ec.h"

typedef struct Student {
    char *name;
    int   age;
} Student;


EC(Student)


int main(int argc, char *argv[])
{
    StudentQueue *student_queue = Student_Queue();

    Student *student1 = Student_New();
    Student *student1 = Student_New();

    StudentQueue *student_queue = Student_Queue();

    Student_Enqueue(student_queue, student1);
    Student_Enqueue(student_queue, student2);

    Student *st;
    for_queue(student_queue, st) {
        printf("%s age is %d", st->name, st->age);
    }

     return 0;
}
```

EC Stack

EC Stack serves variables last serves first nature. Good for sequential access rather than random accesses and also reversing variables order.

Syntax overview

```
#include "ec.h"

typedef struct Student {
    char *name;
    int   age;
} Student;


EC(Student)


int main(int argc, char *argv[])
{
    StudentStack *student_stack = Student_Stack();

    Student *student1 = Student_New();
    Student *student1 = Student_New();

    StudentStack *student_stack = Student_Stack();

    Student_Push(student_stack, student1);
    Student_Push(student_stack, student2);

    Student *st;
    for_stack(student_stack, st) {
        printf("%s age is %d", st->name, st->age);
    }

     return 0;
}
```

EC String

EC String Array

Syntax overview

```
#include "ec.h"

typedef char* Name;

EC(Name)

int main(int argc, char *argv[])
{
    NameArray *name_list = Name_list();

     Name *name1 = Name_New();
     Name *name2 = Name_New();

    name_array->var[0] = (Name) "Arnold";
    name_array->var[1] = (Name) "Silvester";

    Name *name;
    for_array(name_array, name) {
        printf("%s\n", *name);
    }

     return 0;
}
```

EC String List

Syntax overview

```
#include "ec.h"

typedef char* Name;

EC(Name)

int main(int argc, char *argv[])
{
    NameList *name_list = Name_List();

    Name *name1 = Name_New();
    Name *name2 = Name_New();

    *name1 = (Name) "Arnold";
    *name2 = (Name) "Silvester";

    Name_Append(name_list, name1);
    Name_Append(name_list, name2);

    Name *name;
    for_list(name_list, name) {
        printf("%s\n", *name);
    }

     return 0;
}
```

References:

If you are a newbie to C language download and read the `C Basics' booklet from the link below.

http://www.eclab.org/doc/c_basics

or

If you need to remind C language language back in quick way download `Back to C' from the given link below.

http://www.eclab.org/doc/back_to_c

or

If you want to learn C step by step from easy way you can buy our courses from the links given below. Arguably say It's a tremendous support for us develop EC to the next level.