structure in c

structure in c language ->

1. structure is a way to group variables
2. structure is a collection of dissimilar elements
3. defining structure means creating new data type

as per the function of the c language data is of 4-5 types as int, float, char , double but in real life data type can be fir more than that infact data ki category hum jis system ke leye software bana rahe hai us ke upar depend karta hai

as we know data type or of two type

1. primitive data type - int,char ,float,double

whatever variable we make by using primitive data type by this we can store only single value i.e me can store only fundamental data

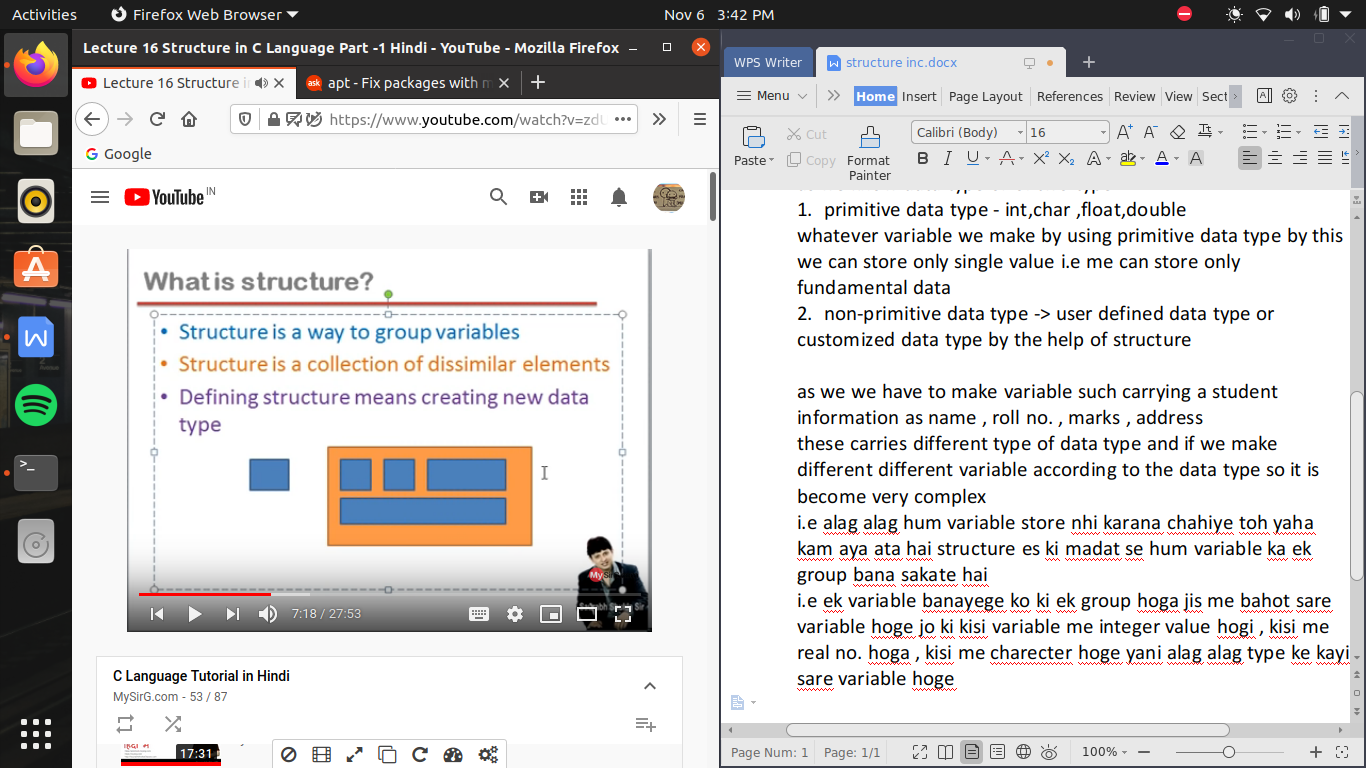
1. non-primitive data type -> user defined data type or customized data type by the help of structure

as we we have to make variable such carrying a student information as name , roll no. , marks , address

these carries different type of data type and if we make different different variable according to the data type so it is become very complex

i.e alag alag hum variable store nhi karana chahiye toh yaha kam aya ata hai structure es ki madat se hum variable ka ek group bana sakate hai

i.e ek variable banayege ko ki ek group hoga jis me bahot sare variable hoge jo ki kisi variable me integer value hogi , kisi me real no. hoga , kisi me charecter hoge yani alag alag type ke kayi sare variable hoge



defining a structure ->

struct <name of data type(i.e form now this will become data type)>

{

//variable declarations here(jo bhi es ke andar data type ban rahe hai )

};

sirf definition se koe memory consume hi nhi hoti jab tak use run na karwaya jaye (or memory variable ko milati hai data type ko nhi toh agar mai varible na banau toh memory consume hi nhi hogi)

/\* store information of date \*/

#include<stdio.h>

void main()

{

}

struct date

{

int d,m,y;

};

struct student

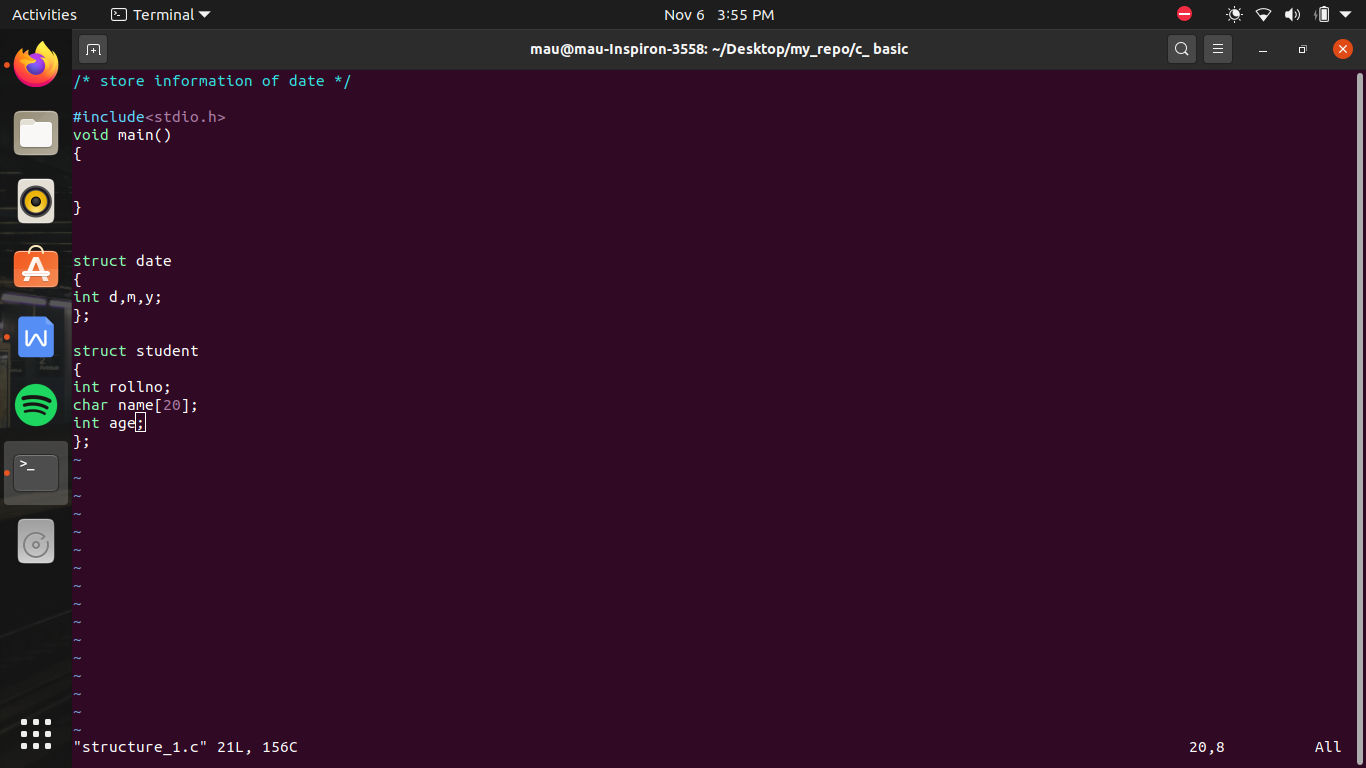
{

int rollno;

char name[20];

int age;

};



this data type we can define either outside the function or also can be inside the function

agar bahr define kiya hai toh kahalayega global definition or pure program me kahi pr bhi use kiya ja sakata hai

jab ki agar andar define kiya ho toh use bolate hai local definition or yeh sirf usi function ke andar use kar sakate hai

how to declare variable ->

/\* store information of date \*/

#include<stdio.h>

struct date

{

int d,m,y;

}d2,d3;

//here we can also make varible like this and this variable will me consiber as global variable ,, //d2 is a variable having data type date

//and each variable of it are consuming 6 bytes of memory

struct date d1;

//we can also make data type outside the function as a global variable which can be use on whole function

//here struct is keyword , date is a data type and d1 is a variable

void main()

{

struct date today; //now here we want gave user the option to assign the value

printf("enter today's date \n");

scanf("%d/%d/%d",&today.d,&today.m,&today.y);

printf("date = %d/%d/%d",today.d,today.m,today.y);

//imporatant as we writing / in between so user also have to sepearate is by / while entering input

}

/\*

void main()

{

struct date today,d1;

today.d=06;

//if we want to initialize the variable later as because we didn't make d variable independently so we have to use dot operation

today.m=11;

today.y=2020;

d1.d=today.d; //here copy the data present in variable today to another variable d1 here we do manually assign each variable we can do this work as like we do in array or strings

d1.m=today.m;

d1.y=today.y;

d1=today; //as like array data is assign as same sequence as today have bacause both the varauible is of same data type date ,so that is why it is possible to assign the value directlty

}

\*/

/\*

void main()

{

struct date today={6,11,2020};

//we can initialize the value as at the time of definition also as same as we initialize the array , in currly bracket separated by commas

//or yaha pr bhi assignation of value is happen in sequence as the sequence we write at the time of defining

//or agar defininf me alag alag type ke data type hote toh hame wohi order fallow karana padeaga jaise ki structure bananya gaya hai

}

\*/

/\*

void main()

{

int x;

//here we use data type and made a variable as like this we make varaible of non primitive data type

struct date today;

//here we use date as a data type and make variable today

//but ek rule hota hai jab bhi use defined data type use karate hai toh us ke pahale struct key word likhana jaruri hai

}

\*/

/\*struct date

{

int d,m,y;

}d2,d3;

\*/

//here we can also make varible like this and this variable will me consiber as global variable ,, //d2 is a variable having data type date

//and each variable of it are consuming 6 bytes of memory

struct student

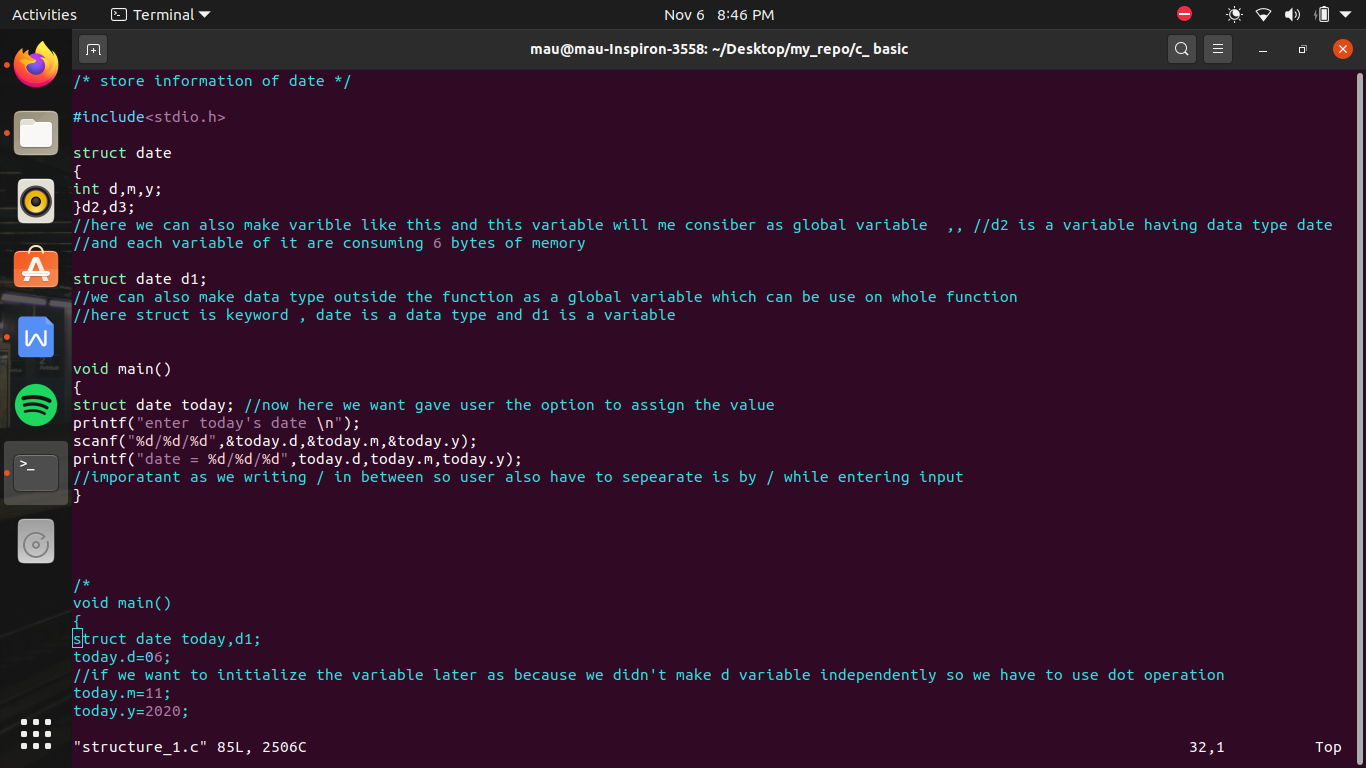
{

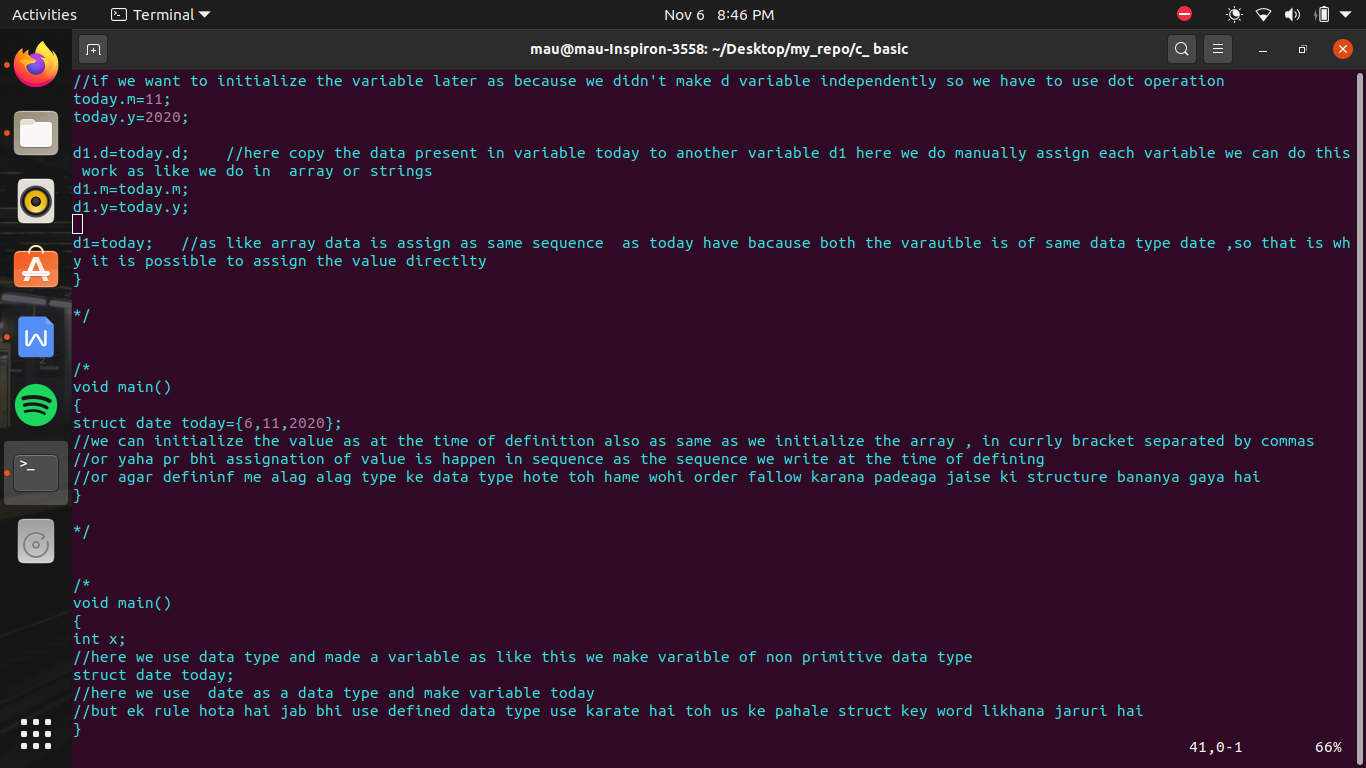
int rollno;

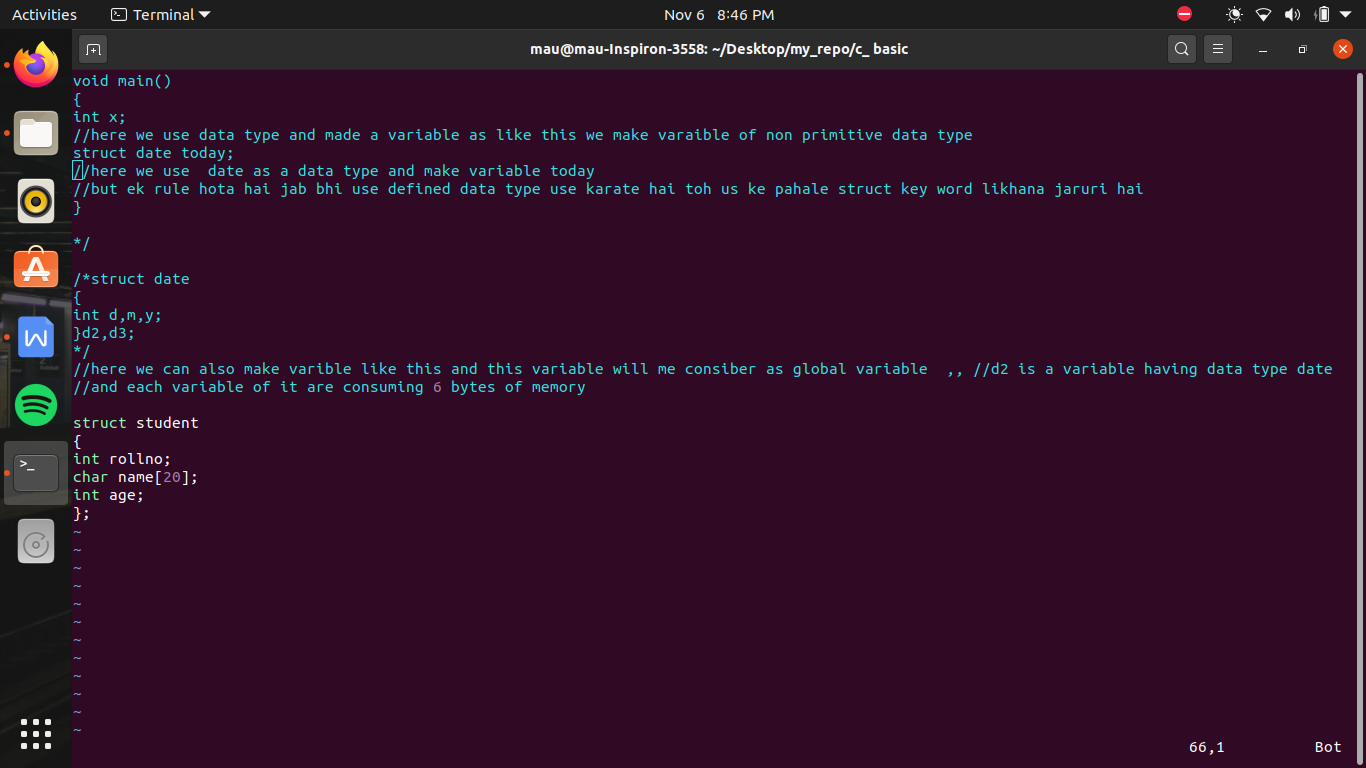
char name[20];

int age;

};







/\* we make a book \*/

#include<stdio.h>

struct book

{

int bookid;

char title[20];

float price;

};

struct book input() //input value is going to assign in b1 and b1 having data type book so input also return struct

{

struct book b;

printf("enter bookid ,title ,price \n");

scanf("%d",&b.bookid);

//as scanf and gets both take data as as input and this value is taken by input buffer , so agar input buffer bhara hua hoga toh yeh fresh input nhi magega

//yaha hum pahale input buffer khali hai or jab hum ne ese input dediya toh yeh bhar gaya toh ab jab yeh gets pr ayega toh input nhi magega ,

//or gets ki help se buffer khali ho kr scanf ("%f") chal jayega toh ab ese ke solution ke leye we use

//fflush(stdin) toh yeh gets function keleye buffer khali kar dega

//fflush is used to clear the buffer

// and accept the next string

fflush(stdin);

gets(b.title);

scanf("%f",&b.price);

return(b);

}

void display(struct book c)

{

printf("\n%d %s %f\n ",c.bookid,c.title,c.price);

}

void main()

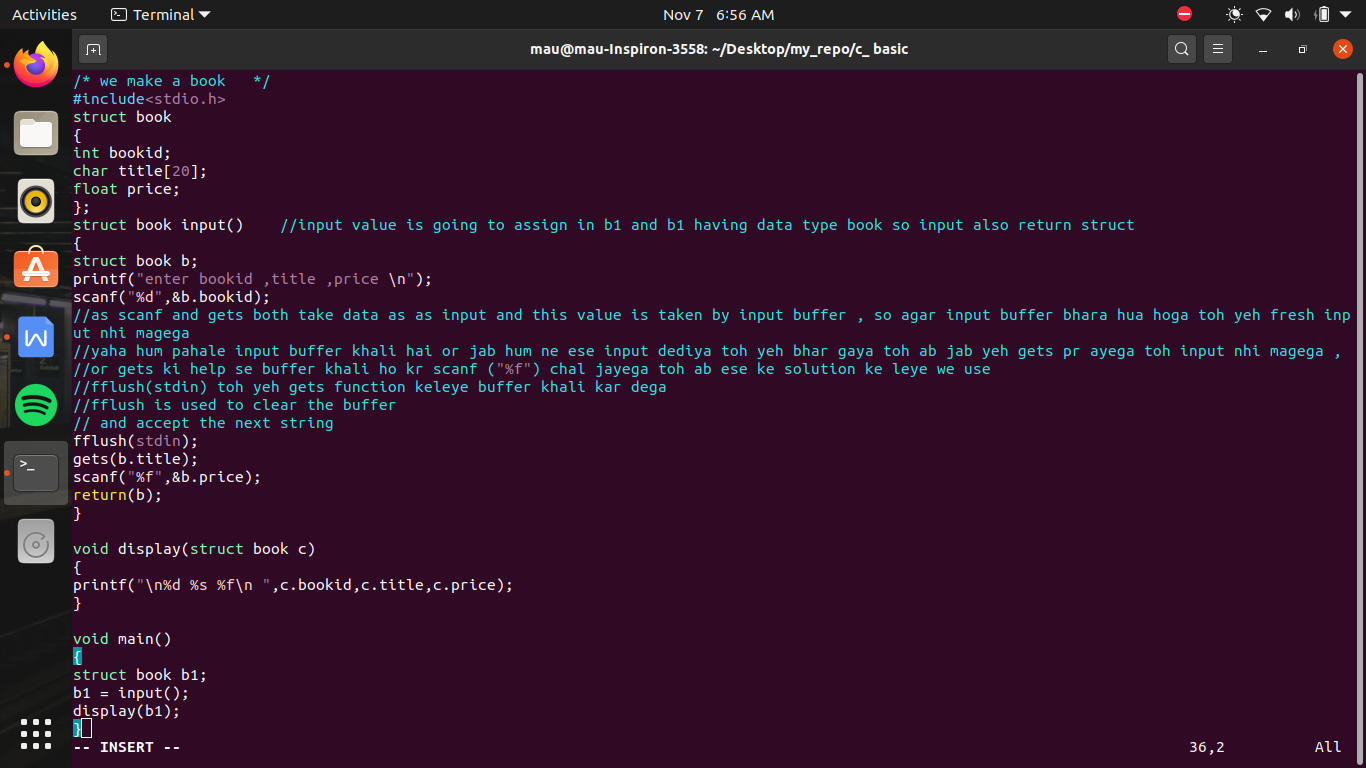
{

struct book b1;

b1 = input();

display(b1);

}



dynamic memory allocation ->

SMA - static memory allocation

DMA - dynamic memory allocation

/\* types of memory allocation \*/

#include<stdio.h>

void main()

{

int a; // here a is the example of static memory allocation

//static memory allocation means at the time of compilation memory kitani consume hogi variables ke nam pe es ka decision ho jana

//i.e decleration statement ke zariye jo varaible banate hai wo SMA ke example hai

float b; //SMA

char x[10]; // SMA

//SMA hi kyu kahate hai because at the time of compilation hame pata chl jata hai when ever be the program will run it will take that amount of memoryand also the time and the destroy of memory is also fixed at the time of decleration , so if we want any memory in between so this we cant do because destroy hona pahale se hi tay hai so that is why is called statis memory allocation

}

//DMA - es me run time pe hi decide kiya jayega ki kitani memory allocate honi hai toh compiler ko pahale se nhi paha rahega ki kitani memory allocate hone wali hai run time pe ,DMA wale variable ka kuch nam nhi hota sirf address hote hai or hum sirf address ki help se es access karate hai

//

// agar hame user se data input karwanan hai toh hame pata hai pahle se hi user 10 variable input karega, toh hum pahale se 10 variables bana dege toh yeh statis memory allocation

//or agar hame nhi malum user kitani values enter karata hai or jo bhi wo values enter karata hai hame store karana hai or use run karana hai, so to allocate dynamic memory we call 4 funtions

DMA ->

1. malloc()
2. calloc()
3. realloc()
4. free()

/\* DMA \*/

/\*

{

malloc(); ->

//agar fun es function ko call kar rahe hai toh yeh toh ek action statementhai, decleration statement nhi toh jab compiler ese padega toh use nhi pata chl payega ki kitane variables bana rahe hai ya ni at time of compilation

//malloc function jab run hoga , yani at the time of run tab memory block creat hoga

//

}

\*/

syntax of malloc function

//here we are going to pass an argument which is the size of malloc function

{

float \*p;

p=(float\*) malloc(6);

//i.e here 6 is not the no. of variable it means each variable made by malloc consume 6 bytes i.e we pass the size of the variable

//here we don't tell the data type of the function

//but here hame toh pata hai ki es me kya pass hoga yani , integer ya charecter ya real value

//let say es me real value pass hogi , toh address ko es me se pass karane ke leye hum ek float type ka pointer bana lete hai or use equate kr dete hia malloc(6) block ke

//yani ek 6 bytes ka block banaya malloc se or us ka address p me pass kr diya

\*p=3.4

//by this we assign the value in the varible made by malloc

//malloc always return block type pointer because hame nhi pata hota yeh kon sa data type return kr raha hota hai

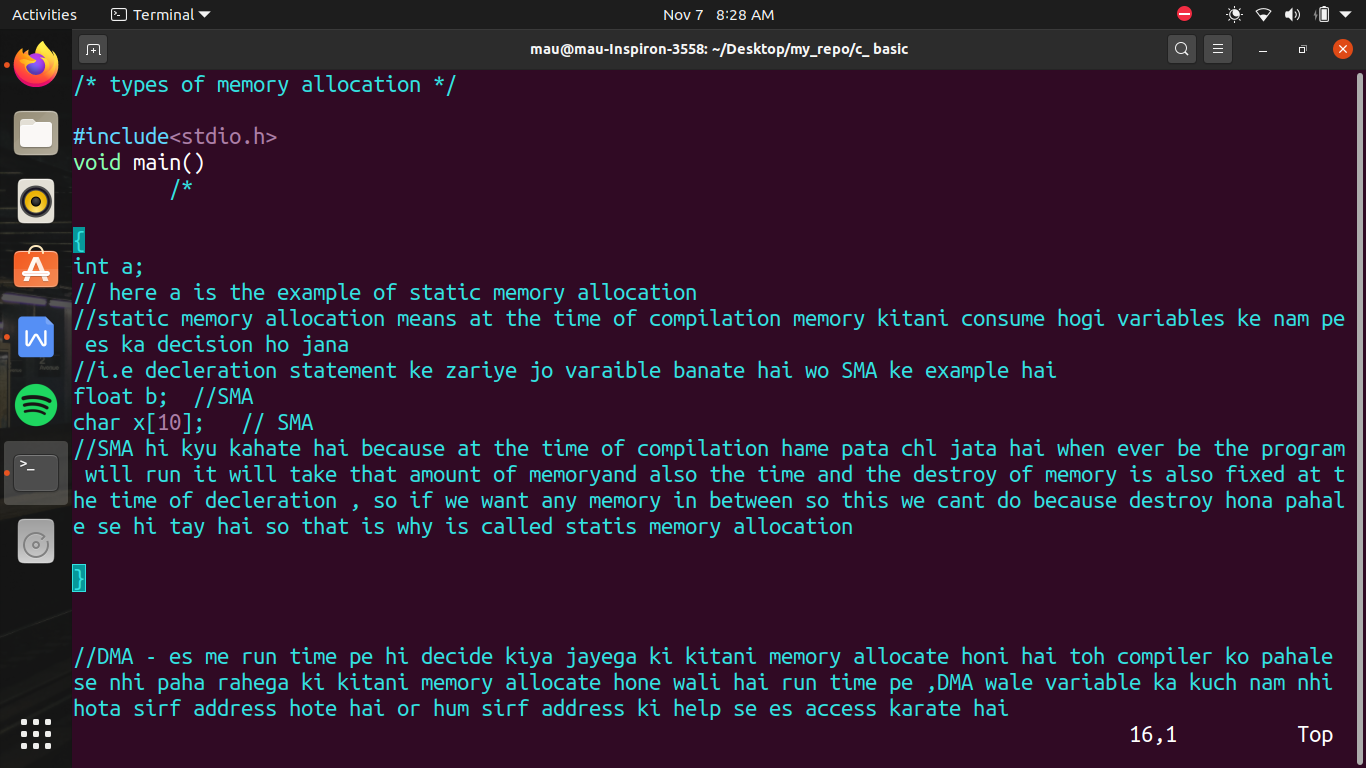
//but yaha malloc is void type pointer and p is float type pointer , asal me hona yahi chahiye ki agar p float type ka pointer hai toh es me float type ka hi address pass hona chahiye so

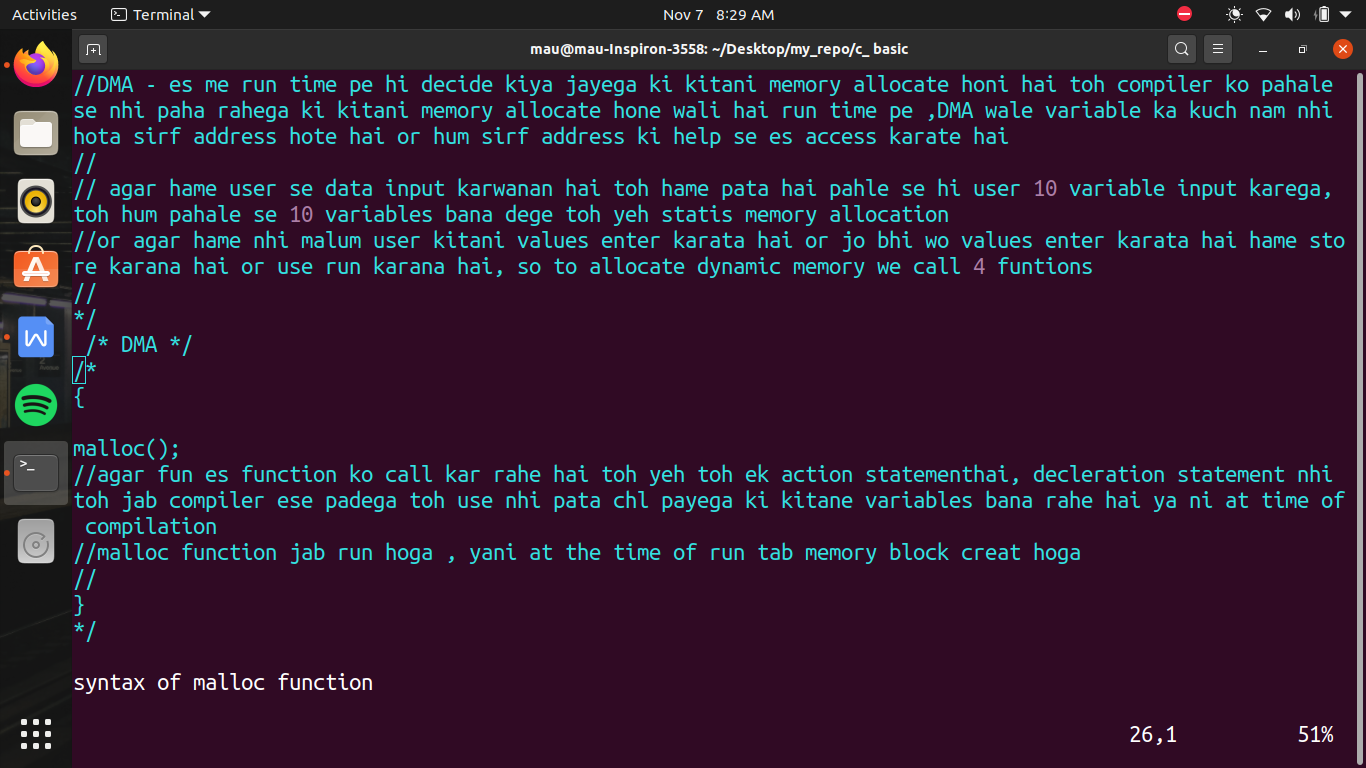
//we do type\_cast the value which came from the malloc

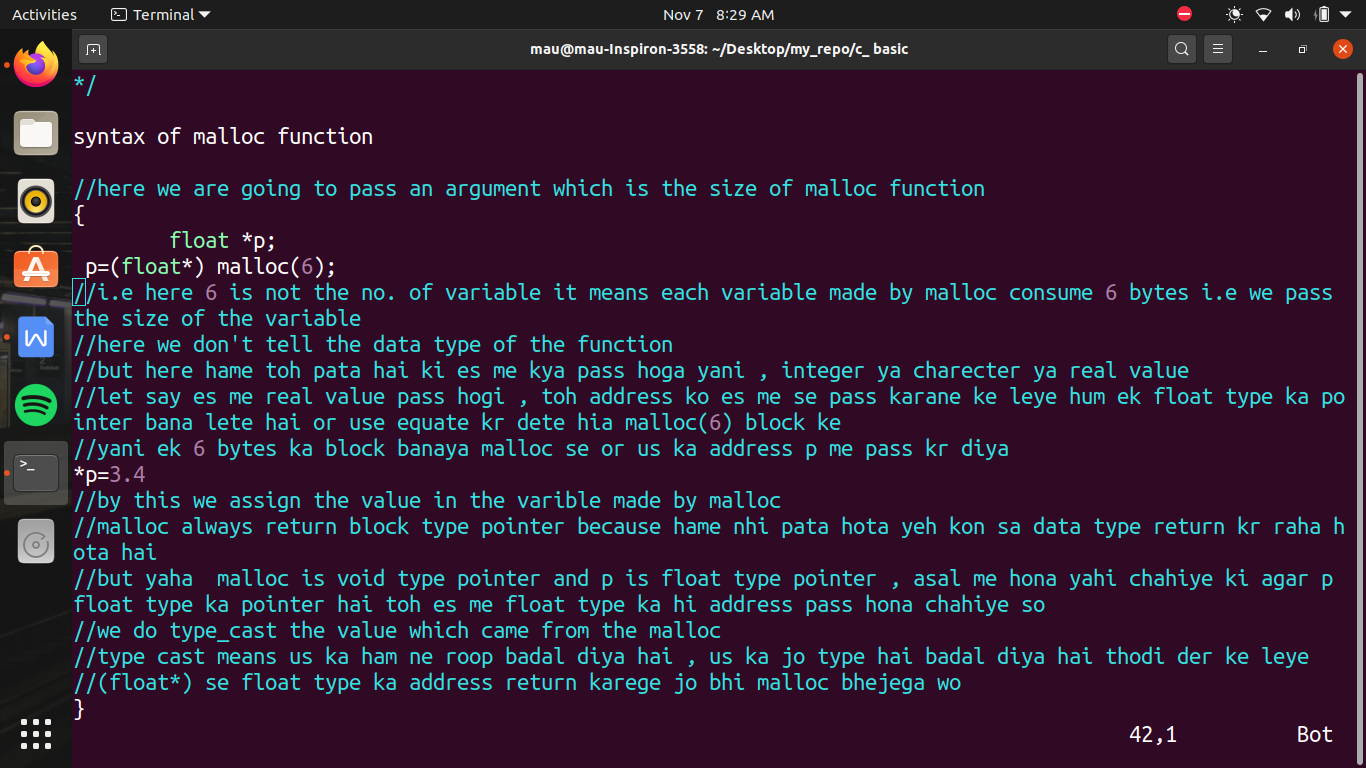
//type cast means us ka ham ne roop badal diya hai , us ka jo type hai badal diya hai thodi der ke leye

//(float\*) se float type ka address return karege jo bhi malloc bhejega wo

}





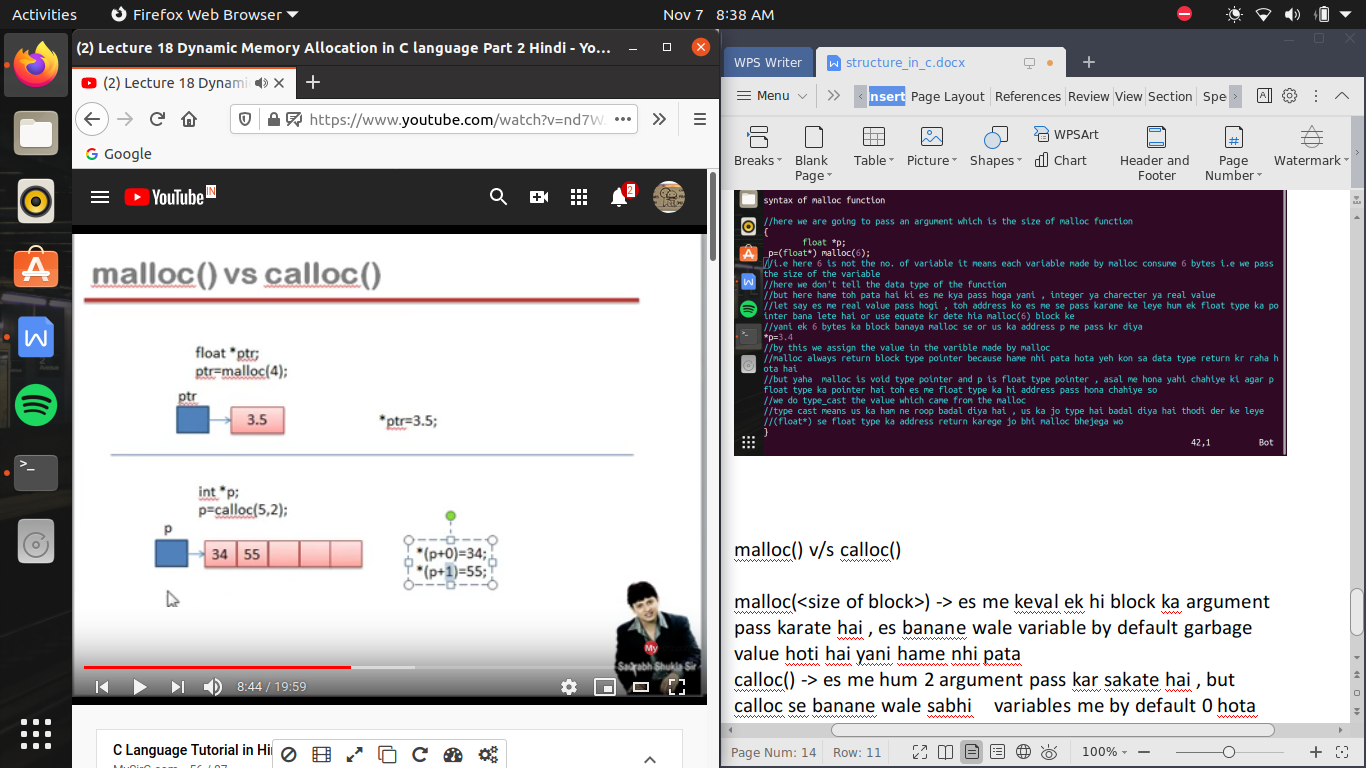


malloc() v/s calloc()

malloc(<size of block>) -> es me keval ek hi block ka argument pass karate hai , es banane wale variable by default garbage value hoti hai yani hame nhi pata

calloc() -> es me hum 2 argument pass kar sakate hai , but calloc se banane wale sabhi variables me by default 0 hota hai

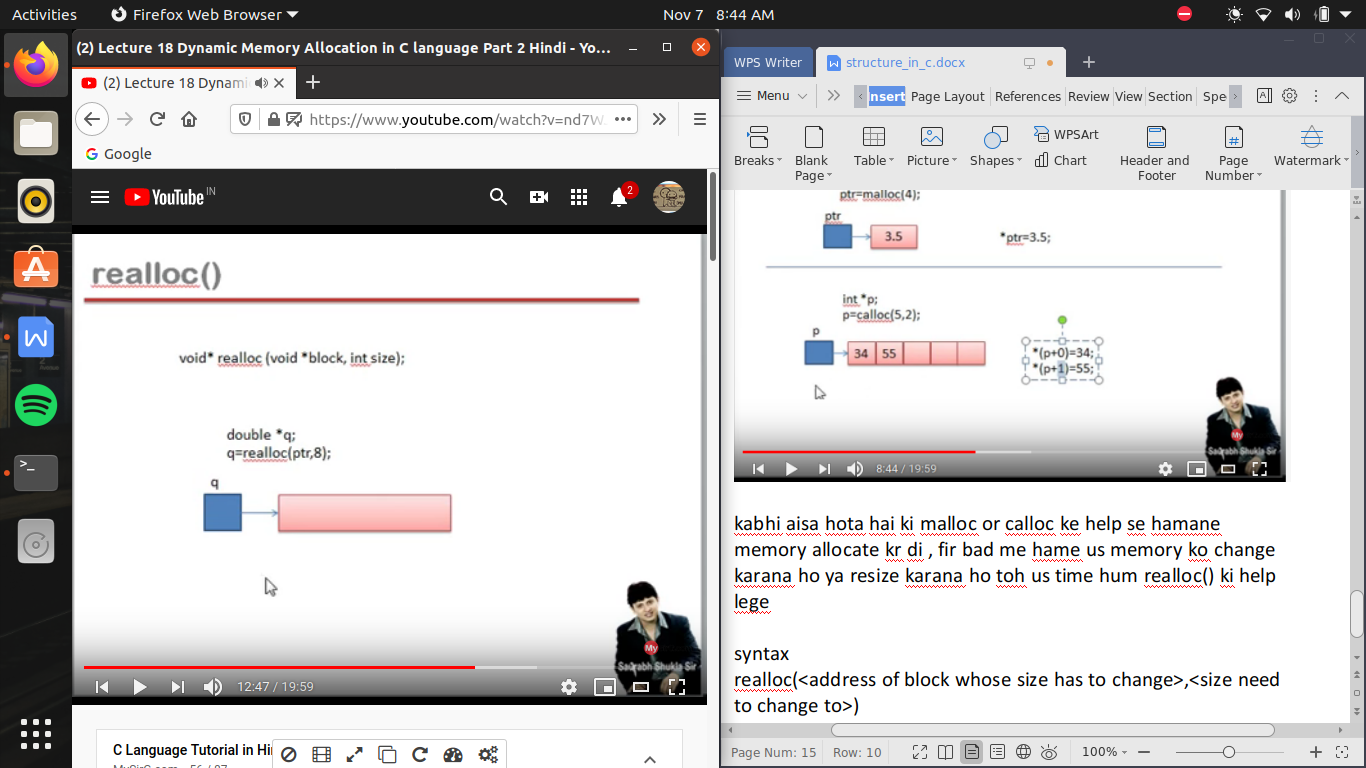
calloc(5,2); means 5 block , carrying 2 bytes each



kabhi aisa hota hai ki malloc or calloc ke help se hamane memory allocate kr di , fir bad me hame us memory ko change karana ho ya resize karana ho toh us time hum realloc() ki help lege

syntax

realloc(<address of block whose size has to change>,<size need to change to>)



free() ->

it realeases the memory

as the memory allocated on variable made my static memory allocation SMA is released as soon as function end

but the memory allocated by using malloc and calloc , memory does not release even after end of function , this memory releases only after end of program

so when we don’t the varible made by malloc or calloc we write

free(<address of the variable which need to free>);

free can never be able to free the memory allocate due to SMA its only works on variable made by malloc and calloc or resize by realloc