**Day 11**

* **free command** is a tool in linux that's displays information about memory usage and swap space on system.
* free [OPTION]

if two free commands are used then double free error is detected

* **strings**

strings is the collection of characters the string should be ended with a null character.

**Syntax :**

char name[]="hello world";

char name[6]="hello";

char name [6]={'h','e','l',l',o,'\0};

char str1[20] // at the time of compilation

//declaration with initialization

char name [][20]={"bhima","shiva",shankar,"havila"};

char Name[10][50]; // there are 10 names each of max capacity of 50 chars

**sample code:**

#include <stdio.h>

int main()

{

char Name[5];

int i;

for(i=0;i<5;i++)

scanf("%c",&Name[i]);

for(i=0;i<5;i++)

printf("\n%c=%d",Name[i],Name[i]);

return 0;

}

**Output:**

havila

h=104

a=97

v=118

i=105

l=108

* **strcpy**

char \*strcpy(char \*dest,const cahr \*src);

char \*strncpy(char \*dest,const cahr \*src,size\_t n);

* **strcat**

char \*strcat(char \*dest,const cahr \*src);

char \*strncat (char \*dest,const cahr \*src,size\_t n);

the length of destination should be greater than the source.

* **strcmp**

strcmp it will check character by character by checking ascii values

whenever the character doesnt match it will stop and doent check the next character.

if both the strings are same it gives 0

if str1 is less than string 2 it gives the negative value

if str1 is greater than string 2 it gives the positive value

* **strcat**

int \*strcat(const char \*s1,const char \*s2);

memcpy() = Copies data from one block of memory to another

memset() = Sets all of the bytes in a block of memory to the same value

strtok() = Splits a string into pieces using delimiters

strxfrm() = Convert characters in a string from ASCII encoding to the encoding of

the current locale

strlen() = Return the length of a string

strerror() = Returns a string describing the meaning of an error code

strstr() = Returns a pointer to the first occurrence of a string in another string

char \*strstr(const char \*haystack const char \*needle);

strrchr() = Returns a pointer to the last occurrence of a character in a string

strtok() = Splits a string into pieces using delimiters

char \*strtok(char \*str, const char \*delim);

char \*strtok\_r(char \*str, const char \*delim, char \*\*saveptr);

**sample code for string comparison:**

#include<stdio.h>

#include<string.h>

int main(){

char s1[20],s2[20];

char d1[40],d2[40];

int ret =0;

scanf("%s%s",s1,s2);

printf("\ns1=%s\ts2=%s",s1,s2);

ret =strcmp(s1,s2);

printf("\nret =%d",ret);

}

**Output:**

havila

Priya

s1=havila

s2=Priya

ret =-8

**sample code:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main(){

char s1[20],s2[20];

char d1[40],d2[40];

char \*ptr = NULL;

int ret =0;

scanf("%s%s",s1,s2);

printf("\ns1=%s\ts2=%s",s1,s2);

ptr=(char \*)malloc(strlen(s1)+1);

/\*

ret =strcmp(s1,s2);

printf("ret =%d",ret);

\*/

ptr=strcpy(d1,s1);

printf("\nd1=%s",d1);

printf("\nptr =%s",ptr);

return 0;

}

**String reverse:**

#include<stdio.h>

#include<string.h>

void reverseString(char str[]) {

int n = strlen(str);

for (int i = 0; i < n / 2; i++) {

char temp = str[i];

str[i] = str[n - i - 1];

str[n - i - 1] = temp;

}

}

int main() {

char str[100];

printf("Enter a string: ");

gets(str); // Note: gets() is unsafe, consider using fgets() in real applications

reverseString(str);

printf("Reversed string: %s\n", str);

return 0;

}

**Output:**

Enter a string: havila

Reversed string: alivah

**structures :**

struct is the keyword

struct tagName

{

members of structure;

};

struct square

{

int length;

int breadth;

};

struct chair

{

int nolegs;

char make [20];

char material[20];

char colour[20];

float price;

char date of manufacturing[0];

char placemanu[20];

};

int a1,a2,a3;

struct square s1,s2,s3;

struct square

{

int length;

int breadth;

}s4,s5,s6; this is like fixing the variables

**can we create arrays of structures**

int a1,a2,a3;

struct square s1,s2,s3,sq[10];

struct square

{

int length;

int breadth;

}s4,s5,s6,s[10];

typef struct square SQR;

SQR s7,s8,s9// declaring the sqr it is a type define

void func(struct square s)//always writing this two words is difficult hence it can be declared as above

{

}

typedef struct square

{

int len;

int breadth;

}SQR1;

SQR s1;

/using the structure we need to access the elements properties or members of the structure

it can be accessed in 2 ways

1. **.** it is used in static variable **namevar.membername**

when s1 is normal variable static variable memory is managed by memory management

2.)  **->** **namevar -> membername**

pointer variable

s1.len// if it is a normal variable

SQR1 \*ptr;

ptr is the pointer variable so to access SQR it should be ptr->len

according to standard ncc only fundamental datatypes and user defined datatypes are used but functions are not possible in structures

structure should be declared globally to above the main hence it is accessible

always remember of structure padding otherwise the structure size will be more

#include<stdio.h>

#include<string.h>

struct emp;

{

int eid;

float esal;

char eaddress[20];

char ename[20];

char egender;

};

typedef struct employee emp;

int main()

{

emp el;

e1.eid =101;

e1.esal=1000000.1;

structcpy(e1.ename,"havila");

strcpy(e1.eaddress,"nyc");

e1.egender='f';

#include<stdio.h>

#include<string.h>

struct employee

{

int eid;

float esal;

char eaddress[20];

char ename[20];

char egender;

};

typedef struct employee emp;

int main()

{

emp e1;

emp e[3];

int i;

e1.eid =101;

e1.esal=1000000.1;

strcpy(e1.ename,"havila");

strcpy(e1.eaddress,"nyc");

e1.egender='f';

printf("\n employee details are:" );

printf("\n id:%d",e1.eid);

printf("\n name:%s",e1.ename);

printf("\n gender:%c",e1.egender);

printf("\n enter employee details\n");

for(i=0;i<3;i++)

{

printf("\n name:");

scanf("%s",e[i].ename);

printf("\n id:");

scanf("%d",&e[i].eid);

printf("\n gender:");

scanf(" ");

scanf("%s",&e[i].egender);

printf("\n address:");

scanf("%s",&e[i].eaddress);

printf("the employee details are :");

printf("\nname:%s",e[i].ename);

printf("\nid:%d",e[i].eid);

}

return 0;

}