**DAY 05**

**OPERATORS**

**Logical operators:**

* && and ||
* && The logical AND operator returns true only if both operands are true; otherwise, it returns false
* || The OR operator returns true if at least one of the operands is true; otherwise, it returns false.

**Process of pushing codes into local repository in git:**

> git clone https://g

Cloning into…..

>cd repo

>CGBatch name

>ls -la

> >cd demo

> >mkdir demo

> >cd demo

> >touch t1.txt

> >ll

> >git add t1.txt

> >it will push the reference to .git folder

> > git commit -m “ type anything”

> if commit is not done we might loses the changes made .with commit every line of command is in local directory .

> >git config --global user.Email”<email.id>”

> > git config user.name”<username>”

> this is used so that we don’t have to give the user name every time

> > git pull origin main (if the code is not in repository and push gets rejected)

> > git push origin main

> pushing to original branch in the sub branch

When using this command it will push the file to git hub account

**Makefile:**

It is a set of commands (similar to terminal commands) with variable names and targets to create object file and to remove them. In a single make file we can create multiple targets to compile and to remove object, binary files. You can compile your project (program) any number of times by using Makefile. The main idea of this file is to specify the dependencies.

**make utility :** This is a command line utility and used to process the instructions written in the Makefile.

First we need to create 4 files in which 2 of them are .c files and 1 header ( .h ) file and 1 make ( .mk ) file.

Make utility -- compile,build,execute as well as clean

To maintain grp of prgrms

28 oct

--------

make utility:

building files or compiling muktiple files:

gcc -o app ./obj/calc.o ./obj/main.o

mv app ./bin

**modify main.c**

gcc -c I ./inc/ ./src/main.c -o ./obj/main.o

gcc -o ./bin/app01 ./obj/calc.o ./obj/main.o

to run: ./bin/app

**Implicit declaration:**u have used the interface but haven't declared the interface name

If we make any changes in calc.c we have to update calc.h and follow the above steps again

We need only three files source, header and lib

We can remove and rebuild bin,obj:

rm ./bin/\*

rm ./obj/\* (These 2 should happen automatically)

make utility fetches for the modified source code... It checks for the updations

If there are n files modified it compiles all the files automatically

And also cleans the files automatically

Makefile:

targetapp: target1 target2 ---- they are .o files

gcc -o .bin/targetapp target1 target 2

target1: dependencies1

gcc -c sourcecodes

target2: dependencies2

gcc -c sourcecodes

#cleaning ---># commenting

echo cleaning now

rm -rf ./bin/\*

rm -rf ./obj/\*

echo cleaning done

To run : make -f ./scripts/makefile (From rootfile)

make (from scripts)

area.c

code coverage tool :gcov

-------------------------

which line of code executed and how many times

gcc -fprofile-arcs -ftest-coverage <file1.c><file2.c> ---> gcov files will get created

file1.gcno

when we run it we get file1.gcda

gcov file1.c -> it will give clear report

Static Analysis using splint

-----------------------------

uninitalized pointers are wild pointers---very dangerous as they can point to anywhere

it might crash prgrm or give garbage value

warns potential bugs

valgrind--dynamic memory tools opp to splint--- a static memory tool

|

run time

Error: Segmentation fault (core dumped) (Eg house construction without proper land/address collapses)