**UNX511 LAB2**

**ANDRE LUIZ VALLE ROSA – 115997173**

**Makefile to build the Library pidUtil.a**

CC=g++ # compiler g++

CFLAGS=-I # add rearch for .h files

CFLAGS+=-Wall # add extra warnings on compilation

CFLAGS+=-c # option for incremental compilation when using multiple files

AR=ar # ar is the option to maintain file archinves to add/delete members of an archive

# build command for compilation as g++ -I -Wall -c pidUtil.cpp -o pidUtil.o

# -o option means the output will be an object file to be used later

pidUtil: pidUtil.cpp

$(CC) $(CFLAGS) pidUtil.cpp -o pidUtil.o

# compile the object file into a library .a

# option rcs creates the archive, insert files and write object file index

lib: pidUtil.o

$(AR) rcs libPidUtil.a pidUtil.o

# removes .o and .a files to prepare for a new build

clean:

rm -f \*.o \*.a

# copies the new lib to a specified folder

install:

cp libPidUtil.a ../.

cp pidUtil.h ../.

#when "make all" forllow the script pidutil and lib to completely build the library

all: pidUtil lib

**Makefile for the Lab2.cpp**

CC=g++ # Select Compiler (g++)

CFLAGS=-I # Add search for .h header files

CFLAGS+=-Wall # Includes extra warnings during the compilation process

FILES=lab2.cpp # File names to be compiled

LIBS = -L. -lPidUtil # -L adds search for libraries, -L. lib will be in the same folder, then add lib name

# build command as 'g++ -I -Wall lab2.cpp -L. -lPidUtil'

# -o indicates that the output is Lab2 (-o Lab2)

Lab2Test: $(FILES)

    $(CC) $(CFLAGS) $(FILES) -o Lab2 $(LIBS)

# force erases all .o files and the executable for a new clean compilation

clean:

    rm -f \*.o lab2

all: Lab2Test

**Lab2.cpp code**

// UNX511 lab2

// author: Andre Rosa

// compile: g++ -I -Wall -o <appName> <fileName.cpp> -L. <libraries.a>

// example: g++ -I -Wall -o Lab2 lab2.cpp -L. -lPidUtil

// ATTENTION the -L. tells the compiler the libray is in the same folder of the .cpp code

#include "pidUtil.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main (void) {

ErrStatus errorStatus;

string errorMsg = " ";

string name = "no name";

vector<int> pidList;

//-------------------------------

// 1. Print all PIDs and Names

errorStatus = GetAllPids(pidList);

if (errorStatus == 0) {

for ( auto i: pidList ) {

errorStatus = GetNameByPid(i, name);

if (errorStatus == 0) {

printf("%d %s\n", i, name.c\_str());

} else {

errorMsg = GetErrorMsg(errorStatus);

}

}

} else {

errorMsg = GetErrorMsg(errorStatus);

}

//-------------------------------

//-------------------------------

// 2. Set PID to 1 and print PID name

int lPID = 1;

errorStatus = GetNameByPid(lPID, name);

if (errorStatus == 0) {

printf("The name of pid of %d is %s\n", lPID, name.c\_str());

} else {

errorMsg = GetErrorMsg(errorStatus);

}

//-------------------------------

//-------------------------------

//3. Set name to lab2 and print PID

name = "Lab2";

errorStatus = GetPidByName(name, lPID);

if (errorStatus == 0) {

printf("The pid of %s is %d\n", name.c\_str(), lPID);

} else {

errorMsg = GetErrorMsg(errorStatus);

}

//------------------------------

//-----------------------------

//4. Set an invalid name get an error

name = "Lab22";

errorStatus = GetPidByName(name, lPID);

if (errorStatus == 0) {

printf("The pid of %s is %d\n", name.c\_str(), lPID);

} else {

errorMsg = GetErrorMsg(errorStatus);

printf("No such name\n");

}

//----------------------------

return 0;

}