**Practica 16**

Andres Aguirre Gonzalez A01039656

Codigo:

// Andres Aguirre Gonzalez A01039656

// Practica 17

// 7/11/2019

#include <stdio.h>

#include <pthread.h>

#include <time.h>

#include <string.h>

#include <stdlib.h>

char \*task[1000];

char \*dependencies[1000][1000];

int duration[1000];

int started[1000];

int finished[1000];

int numberOfFinished;

int numberOfDependencies[1000];

int numberOfTasks;

void \*runTask(void \*threadid) {

long tid;

time\_t start = time (NULL);

tid= (long)threadid;

printf ("Running task %s\n", task[tid]);

while (time (NULL)-start<duration[tid]);

printf ("Task %s finished\n",task[tid]);

finished[tid] = 1;

numberOfFinished++;

pthread\_exit (NULL);

}

int getIndex(char \*dependency) {

int i;

for (i = 0; i < numberOfTasks; i++) {

if (strcmp (task[i],dependency) == 0) {

return i;

}

}

return -1;

}

int checkDependencies(int taskNumber) {

int i;

for (i = 0;i < numberOfDependencies[taskNumber]; i++) {

if (!finished[getIndex (dependencies[taskNumber][i])]) {

return 0;

}

}

return 1;

}

int main() {

FILE\* file = fopen("table.txt","r");

numberOfTasks = 0;

task[numberOfTasks] = (char \*)malloc (100);

while (fscanf (file,"%s",task[numberOfTasks])==1) {

fscanf (file,"%d",&duration[numberOfTasks]);

char space;

numberOfDependencies[numberOfTasks]=0;

do {

dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]] = (char\*) malloc (100);

fscanf file,"%s%c",dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]],&space);

if (dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]][0]=='-') break;

else if (dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]][strlen (dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]])-1] == ',') {

dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]][strlen (dependencies[numberOfTasks][numberOfDependencies[numberOfTasks]])-1] = '\0';

}

numberOfDependencies[numberOfTasks]++;

}while (space==' ');

task[++numberOfTasks] = (char\*)malloc (100);

}

numberOfFinished = 0;

pthread\_t threads[numberOfTasks];

while (numberOfFinished < numberOfTasks) {

long i;

int rc;

for (i = 0; i < numberOfTasks; i++) {

if (!started[i] && checkDependencies (i)) {

started[i] = 1;

rc = pthread\_create (threads + i, NULL, runTask, (void \*) i);

}

}

}

return 0;

}

Evidencia:

