I OBJETIVOS

En esta práctica el alumno:

- Utilizará un sistema de paso de mensajes como una herramienta para la sincronización y comunicación de procesos concurrentes.

II BIBLIOGRAFÍA

- William Stallings, “SISTEMAS OPERATIVOS”,Prentice Hall.
- Silberschatz, Galvin, Gagne, “SISTEMAS OPERATIVOS”, Mc Graw Hill.
- Neil Matthew & Richard Stones, “BEGINNING LINUX PROGRAMMING”, Wrox.

III RECURSOS

- Una estación de trabajo con Linux con su ambiente gráfico.
- Un editor de texto.
- El compilador de C GNU.

IV ACTIVIDADES

1 Serie de Mercator

En matemáticas, la serie de Mercator o serie de Newton–Mercator es la serie de Taylor del logaritmo natural:

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots.$$

Escrita utilizando notación sumatoria:

$$\ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n} x^n.$$

La serie converge al logaritmo natural (desplazado en 1) cuando $-1 < x \leq 1$.

2 Problema a resolver

El programa que se muestra a continuación realiza el cálculo de $ln(1+x)$ donde $x = 1$ por medio de la serie de Mercator. Para hacer este cálculo de manera precisa, la sumatoria se realiza con 200,000 términos, lo cual implica mucho trabajo de procesamiento. Para que en este trabajo se aproveche la capacidad que brinda un procesador Multicore, éste se reparte en 4 procesos que pueden ejecutarse en paralelo.

El programa que se muestra en la Figura 1 crea un proceso que llamaremos proceso maestro, y cuatro procesos que serán los procesos esclavos quienes realizan los cálculos. El programa principal crea 5 procesos que son el maestro y 4 esclavos. El proceso maestro establece el valor de una bandera que les indica a los procesos esclavos que pueden iniciar los cálculos, y mientras los esclavos hacen el trabajo el proceso maestro esperará a que estos terminen. Todos los procesos se comunican y sincronizan a través de variables en memoria compartida.

Aunque esta solución funciona, esta no es eficiente ya que tanto el maestro como esclavos tienen espera ocupada. Además, es probable que en algunas ejecuciones exista un error cuando se realiza el conteo de procesos.

Ahora hay que utilizar un sistema de paso de mensajes para arreglar el código de manera que genere espera ocupada. La intención es lograr tanto la sincronización como la comunicación con mensajes por lo que es posible prescindir de la utilización de memoria compartida.

```
/*
        Para compilar incluir la librería  m (matemáticas)

        Ejemplo:
            gcc -o mercator mercator.c  -lm
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <sys/time.h>
#include <sys/wait.h>
#include <sys/shm.h>

#define NPROCS 4
#define SERIES_MEMBER_COUNT 200000
```

```
double *sums;
double x = 1.0;
int *proc_count;
int *start_all;
double *res;

double get_member(int n, double x)
{
    int i;
    double numerator = 1;

    for(i=0; i<n; i++ )
        numerator = numerator*x;

    if (n % 2 == 0)
        return ( - numerator / n );
    else
        return numerator/n;
}

void proc(int proc_num)
{
    int i;
    while(!(*start_all));
    sums[proc_num] = 0;
    for(i=proc_num; i<SERIES_MEMBER_COUNT;i+=NPROCS)
        sums[proc_num] += get_member(i+1, x);

    (*proc_count)++;

    exit(0);
}

void master_proc()
{
    int i;

    sleep(1);
    *start_all=1;

    while (*proc_count != NPROCS) {}   // busy wait until all threads are done
with computation of partial sums
```

```
    *res = 0;
    for(i=0; i<NPROCS; i++)
        *res += sums[i];

    exit(0);
}

int main()
{
    int *threadIdPtr;

    long long start_ts;
    long long stop_ts;
    long long elapsed_time;
    long lElapsedTime;
    struct timeval ts;
    int i;
    int p;
    int shmid;
    void *shmstart;

    shmid = shmget(0x1234,NPROCS*sizeof(double)+2*sizeof(int),0666|IPC_CREAT);
    shmstart = shmat(shmid,NULL,0);
    sums = shmstart;
    proc_count = shmstart + NPROCS * sizeof(double);
    start_all = shmstart + NPROCS * sizeof(double) + sizeof(int);
    res = shmstart + NPROCS * sizeof(double) + 2 * sizeof(int);

    *proc_count = 0;
    *start_all = 0;

    gettimeofday(&ts, NULL);
    start_ts = ts.tv_sec; // Tiempo inicial

    for(i=0; i<NPROCS;i++)
    {
        p = fork();
        if(p==0)
            proc(i);
    }
```

```
    p = fork();
    if(p==0)
        master_proc();

    printf("El recuento de ln(1 + x) miembros de la serie de Mercator es
%d\n",SERIES_MEMBER_COUNT);
    printf("El valor del argumento x es %f\n", (double)x);

    for(int i=0;i<NPROCS+1;i++)
        wait(NULL);

    gettimeofday(&ts, NULL);
    stop_ts = ts.tv_sec; // Tiempo final
    elapsed_time = stop_ts - start_ts;

    printf("Tiempo = %lld segundos\n", elapsed_time);

    printf("El resultado es %10.8f\n", *res);
    printf("Llamando a la función ln(1 + %f) = %10.8f\n",x, log(1+x));

    shmdt(shmstart);
    shmctl(shmid,IPC_RMID,NULL);
}
```

Figura 1.- Cálculo de la serie de Mercator con procesos

Importante: Esta solución se tiene que hacer con procesos y emplear los mecanismos de comunicación y sincronización de procesos que aquí mismo se indican, no se aceptarán soluciones que utilicen hilos.

V ENTREGA Y EVALUACIÓN

No incluya líneas de código en sus programas de las cuales desconozca su funcionamiento. El código no conocido será anulado en el funcionamiento de la práctica.

Aunque en semestres anteriores se han realizado prácticas similares a esta, hay aspectos que hacen que esta sea diferente. Cualquier evidencia que muestre el intento de entregar una práctica de semestre anterior será calificada como plagio.

1 Entrega y Revisión

Entregar en el apartado correspondiente de Canvas los programas fuentes con su respectivo `Makefile` en un archivo ZIP.

2 Equipos

Esta práctica se hará en equipos (máximo 2 integrantes), es necesario que en la revisión esté el equipo completo ya que el integrante que no se presente no tendrá calificación en la práctica.

Importante: Al indicarse que el trabajo debe ser desarrollado por equipos, se entiende que no se permite colaboración entre equipos, cualquier evidencia de esto será considerada plagio.

3 Evaluación

Puntualidad en las revisiones	El equipo estuvo completo y puntual en todas las sesiones de revisión.		Si hubo dos o más sesiones con el equipo, el equipo estuvo completo y puntual en casi todas las sesiones de revisión		Si solo hubo una sesión de revisión, el equipo no estuvo completo o no fue puntual. Si fueron dos o más sesiones de revisión, en más de una sesión el equipo no estuvo completo o fue puntual
	+5		+2.5		0
Especificaciones de entrega	La entrega del producto cumple con todas las especificaciones indicadas en el documento de la práctica, por ejemplo, los archivos se entregan de acuerdo a las formas indicadas en el documento de la práctica.				La entrega del producto no cumple con al menos una de las especificaciones indicadas en el documento de la práctica
	+5				0
Funcionamiento	El producto cumple con todas las especificaciones indicadas en el documento y no tiene fallas	El producto muestra una falla no esperada.	Se da una de las siguientes condiciones: -Está incompleto (falta máximo aprox 50%), pero lo demás puede funcionar bien. -Está completo pero muestra dos fallas	Se da una de las siguientes condiciones: -Está completo pero muestra 3 fallas o más. -Está incompleto (falta máximo aprox 50%) y además muestra fallas -Está incompleto (falta máximo aprox 66%)	El producto no funciona o está incompleto (más del 66%).
	+80	+60	+40	+20	0
Interfaz con el usuario	El producto funciona y pudo ser utilizado sin necesidad de recibir indicaciones por el desarrollador, tiene instrucciones claras para ser utilizado.		El producto funciona, pero hubo necesidad de recibir alguna indicación para su uso por parte del desarrollador del producto		El producto carece de instrucciones claras para ser utilizado y requiere que alguno de los desarrolladores esté presente para su utilización o no puede utilizarse debido a que no está completo
	+5		+3.5		0
Claridad en el código	El código es claro, usa nombres de variables adecuadas, está debidamente comentado e indentado. Puede ser entendido por cualquier otra persona que no intervino en su desarrollo.		El código carece de claridad, puede ser entendido por cualquier persona ajena a su desarrollo pero con cierta dificultad.		El código carece de comentarios, está mal indentado, usa nombres de variables no adecuadas.
	+5		+3.5		0
Defensa del producto	Todos los que presentan la práctica son capaces de explicar cualquier parte del producto presentado		Uno de los que presenta la práctica muestra dudas sobre alguna parte del desarrollo del producto presentado		Más de un integrante no muestra evidencia de que conoce el producto, o si el trabajo fue individual, el desarrollador duda sobre el desarrollo del producto que presenta.
	x 1(puntos se multiplican por 1)		x 0.5 (puntos se multiplican por 0.5)		x 0 (puntos se multiplican por 0)
Sobresaliente 20 %	Tiene 1 en todos los puntos anteriores. El producto entregado es sobresaliente, muestra tener la calidad para ser expuesto como un producto representativo de la carrera Hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado				No tiene 1 en todos los puntos anteriores, o el producto entregado no es sobresaliente y no muestra tener la calidad para ser expuesto como un producto representativo de la carrera o no hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado
	+20				0