INFORME

EXAMEN PARCIAL

ESTUDIANTE:

JOEL ROBERT URDAY JERÍ

CURSO:

SISTEMAS OPERATIVOS

2021

En el siguiente informe se va a desarrollar y explicar de forma breve la solución al problema de los Filósofos comensales. Además, sobre el Deadlock y si existe una posible solución a este.

Código

- Esta clase consiste en el descanso de los filósofos.

```
class Chopstics
{
  public:
  Chopstics(){;}
  mutex mu;
};
```

- Aquí se encarga en el tiempo que van a demorar los filósofos comiendo.

```
class tiempo
{
  public:
  int time;
  auto endT(float timed){
  }
};
```

- En el *main* tenemos un *while*, el cual contendrá prácticamente todo el bucle del problema. En el último *auto* de la imagen indica que el filósofo cogió los tenedores y se puso a comer.

```
while (k < com)
{

  int ms = temp.time;
  auto start = chrono::high_resolution_clock::now();

  auto eat = [ms](Chopstics &left_chopstics, Chopstics &right_chopstics, int philosopher_number)
  {

    unique_lock<mutex> llock(left_chopstics.mu);
    unique_lock<mutex> rlock(right_chopstics.mu);

    cout << "Filosofo " << philosopher_number << " está comiendo\n";

    chrono::milliseconds timeout(ms);
    this_thread::sleep_for(timeout);

    cout << "Filosofo " << philosopher_number << " terminó de comer\n";
  };
```

- Los filósofos empiezan a pensar cuando no están comiendo, con ayuda del *thread*. En el último *for* empiezan a comer.

```
Chopstics chp[filosof];
thread philosopher[filosof];

cout << "Filosofo 1 esta pensando\n";
philosopher[0] = thread(eat, ref(chp[0]), ref(chp[filosof - 1]), 1);

for(int i = 1; i < filosof; ++i)
{
  cout << "Filosofo " << (i+1) << " esta pensando\n";

  philosopher[i] = thread(eat, ref(chp[i]), ref(chp[i - 1]), (i + 1));
}

for(auto &ph: philosopher)
{
  ph.join();
}
```

- Aquí indica sobre la cantidad de comida.

```
auto end = chrono::high_resolution_clock::now();
chrono::duration <float> duration = end - start;
temp.endT(duration.count());
k++;
cout<<"Comida sobrante: "<< com - k << endl;
```

- Resultado:

```
Numero de filosofos: 5
Filosofo 1 esta pensando
Filosofo 2 esta pensando
Filosofo 1 esta comiendo
Filosofo 3 esta pensando
Filosofo 4 esta pensando
Filosofo 5 esta pensando
Filosofo 1 termino de comer
Filosofo 2 esta comiendo
Filosofo 2 termino de comer
Filosofo 3 esta comiendo
Filosofo 3 termino de comer
Filosofo 4 esta comiendo
Filosofo 4 termino de comer
Filosofo 5 esta comiendo
Filosofo 5 termino de comer
Comida sobrante: 19
Filosofo 1 esta pensando
Filosofo 2 esta pensando
Filosofo 1 esta comiendo
Filosofo 3 esta pensando
Filosofo 4 esta pensando
Filosofo 5 esta pensando
Filosofo 1 termino de comer
Filosofo 2 esta comiendo
Filosofo 2 termino de comer
Filosofo 3 esta comiendo
```

Implementación en la máquina virtual

Primero crear un archivo .cpp y luego generar el archivo ejecutable mediante el siguiente comando: *g++ -pthread -o "nombre del ejecutable" "nombre del archivo .cpp"*.

```
joel@joel:~/filosofos$ ls -l
total 56
-rw-r--r-- 1 joel joel  1699 may 21 04:31 filo.cpp
-rwxr-xr-x 1 joel joel 49592 may 21 04:31 filosofos
```

Con el comando */usr/sbin/update-rc.d "nombre del ejecutable" defaults* todos pueden acceder al archivo.

Luego accedemos al .profile mediante *vim .profile* y en la úlitma línea añadimos lo siguiente:

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
/etc/init.d/filosofos start
```

Se reinicia la máquina y al prender debería ejecutarse el código.

Deadlock

En español conocido como “interbloqueo”, el deadlock se refiere a un bloqueo permanente de hilos o ciclos en un sistema concurrente. Para estos no existe una solución general. La mayoría de los deadlock implican una competencia entre varios procesos por varios recursos.

Pueden ser evitados si se sabe cierta información sobre los procesos antes de la asignación de recursos. Para cada petición, el sistema controla si satisfaciendo el pedido entra en un estado inseguro, donde puede producirse el deadlock. De esta forma, el sistema satisface los pedidos de recursos solamente si se asegura que quedará en un estado seguro.

Algunos algoritmos que nos pueden ayudar a evitarlos:

- Algoritmo del banquero.
- Algoritmo de grafo de asignación de recursos.
- Algoritmo de solicitud de recursos.

Colaboraciones:

Colaboraciones en la instalación de una nueva máquina virtual Debian, debido a que el anterior tenía varios problemas relacionados con el bash y otros.

Colaboraciones en algunas partes del código.

- Aron Hurtado
- Elvis Minaya
- Leonardo Deza **(solicitud de revisión de su video)**
- Victor Ranilla
- Carlos Corrales