

Centro Universitario de Ciencias Exactas e Ingeniería



Sistemas Operativos

Jonathan Daniel Requeses Nuñez

Cristian Rafael Romero Chavez

Angel Martin Ramirez Castorena

Colas Múltiples

Ramiro Lupercio Coronel.

216787779

216787507

216788066

Colas Múltiples

Objetivo.

Comprender y analizar la función de las colas múltiples para el manejo de los procesos y generar un programa que lleve a cabo el método.

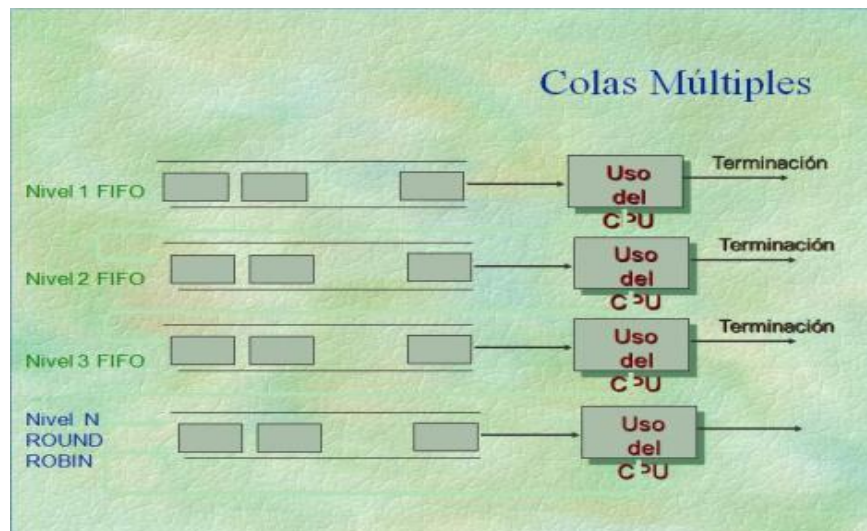
Investigación.

El movimiento de los procesos se determina a través de varias colas de diferentes niveles.

- Un proceso nuevo entra a la red de colas, al final de la primera cola. Se desplaza por FIFO.
- Cuando a un proceso se le termina su quantum de tiempo, se coloca al final de la cola del siguiente nivel.
- El quantum asignado a un proceso cuando pasa a una cola de nivel inferior alcanza un valor mayor.
- Un proceso en cierta cola no puede ejecutarse a menos que estén vacías las colas de los niveles más altos.

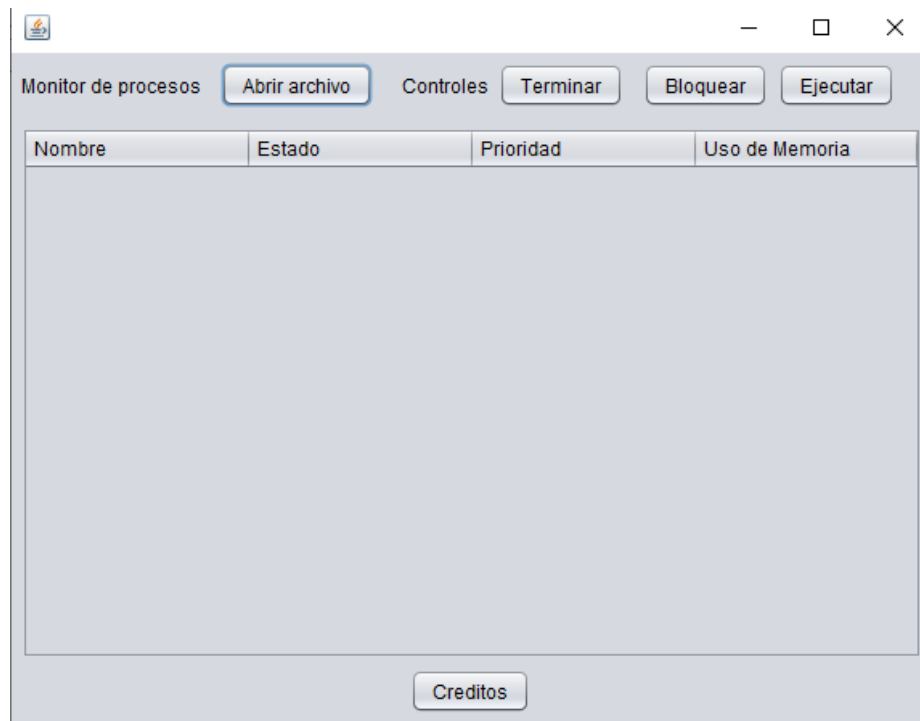
Las colas múltiples están basadas en una pila que sirve como índice de una lista de procesos que se tienen que ejecutar.

Este tipo de algoritmo de planificación se usa para trabajos en batch o de procesamiento por lotes en los cuales se puede saber cual es el tiempo de duración de la ejecución de cada proceso y entonces se puede seleccionar primero el trabajo más corto. El problema que se presenta con éste algoritmo es que los grandes procesos podrían sufrir de inanición dado que cualquier proceso pequeño se “cuela” sobre uno de mayor tamaño y como resultado final se podría dar el caso que el proceso grande nunca obtenga procesamiento.



Desarrollo.

Primero se desarrolló la interfaz con la que se trabajarán los procesos:



En la interfaz como se muestra se cargará un archivo con los datos de los procesos con los que se va a simular el monitor de procesos con la técnica de colas múltiples.

Para dar a notar las colas en el programa, cada una de las colas representa una prioridad de procesos, desde muy alta a baja, cada una se ejecutará después de la otra hasta que todas las colas hayan terminado con sus procesos. A su vez se define en el programa un proceso determinado como proceso padre, desde el cuál se controlarán todos los hilos correspondientes a todos los procesos de cada una de las colas. A continuación se muestra código correspondiente al proceso padre o gestor.

```
public class ProcesoPadre extends Thread{
    public List<Proceso> procesos;
    public List<List<Proceso>> lstPrioridades;
    public DefaultTableModel tabla;
    public JTable procesosTabla;
    public boolean running;
    public int quantum;
    public int numEnEspera;
    public Estado compare;

    public ProcesoPadre(DefaultTableModel _m, JTable _j)
    {
        tabla = _m;
        procesos = new ArrayList();
        procesosTabla = _j;
        running = true;
        quantum = 8000;

        lstPrioridades = new ArrayList<List<Proceso>>();
        lstPrioridades.add(new ArrayList<Proceso>());
        lstPrioridades.add(new ArrayList<Proceso>());
        lstPrioridades.add(new ArrayList<Proceso>());
        lstPrioridades.add(new ArrayList<Proceso>());
    }
}
```

@Override

```
public void run()
{
    //Proceso Ant;
    //numEnEspera = procesos.size();
    if(procesos.size()>0)
    {
        int cont, iLst, numProc, numPri, tProc;
        cont = iLst = 0;
        numPri = lstPrioridades.size();

        while(iLst < numPri)
        {
            List<Proceso> procActuales = lstPrioridades.get(iLst);
            numProc = procActuales.size();
            tProc = 0;
            while(tProc < numProc )
            {
                Proceso proceso = procActuales.get(cont);
                if(proceso.getEstado() == Estado.Nuevo)
                {
                    proceso.start();

                }else if(proceso.getEstado() == Estado.Espera)

                {
                    Proceso p = lstPrioridades.get(iLst).get(cont);
                    p.setEstado(Estado.Ejecucion);
                    p.ejecutarhilo();
                    procesosTabla.setValueAt(p.getEstado().name(), p.id, 1);
                    tProc = 0;
                }else if(proceso.getEstado() == Estado.Bloqueado)
                {
                    tProc = 0;
                    cont++;
                    if(cont == numProc)
                    {
                        cont = 0;
                        continue;
                    }else if(proceso.getEstado() == Estado.Terminado){
                        tProc++;
                        cont++;
                        if(cont == numProc)
                        {
                            cont = 0;
                            continue;
                        }
                    }
                }
                try {
                    Thread.sleep((quantum/procActuales.size()));
                } catch (InterruptedException ex) {
                    Thread.currentThread().interrupt();
                }
            }
        }
    }
}
```

```

        if(proceso.getEstado() == Estado.Ejecucion) {
            Proceso p = lstPrioridades.get(iLst).get(cont);
            p.setEstado(Estado.Espera);
            p.bloquearhilo();
            procesosTabla.setValueAt(p.getEstado().name(), p.id, 1);
        }
        cont++;
        if(cont == numProc)
            cont = 0;
    }
    iLst++;
    quantum = quantum - 2000;
    System.out.println(quantum);
}
}
}

```

En el código anterior se puede ver como se hace el desplazamiento entre las colas, pasando por todos los procesos pertenecientes a la cola de mayor jerarquía y conmutando hacia la siguiente, hasta que todos los anteriores tengan un estado terminado en el monitor, ya que los demás nunca van a poder ser ejecutados hasta que todos los demás estén terminados.

Código de la clase proceso:

```

public class Proceso extends Thread {
    private boolean suspProc;
    private boolean paProc;

    public int id;
    public String name = "";
    public int pid;
    public String session = "";
    public int num_session;
    public String memoria;
    public int segundos;
    public JTable modelo;
    public Proceso ProcesoAnt;
    public Estado estado;
    public int prioridad;
}

```

```

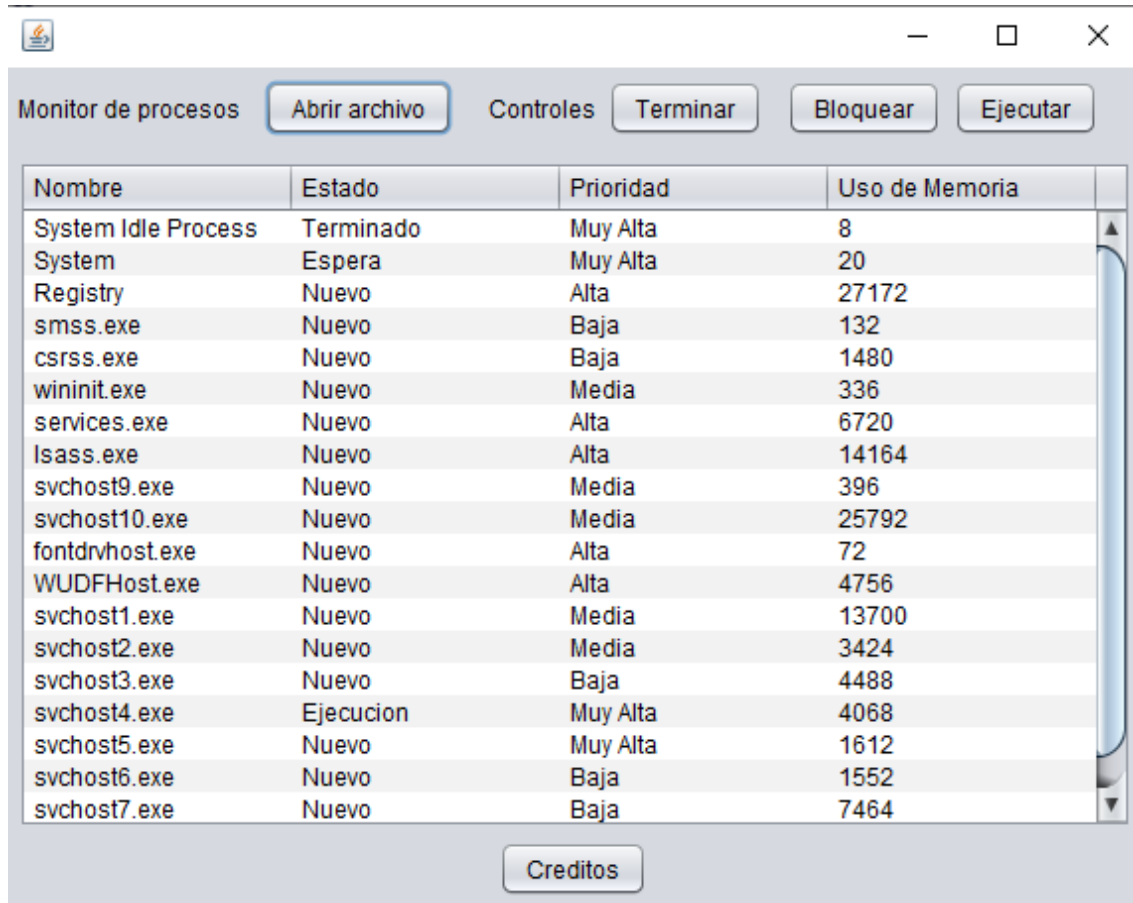
@Override
public void run()
{
    estado = Estado.Ejecucion;
    modelo.setValueAt(estado.name(), id, 1);

    int ciclos = segundos*2;
    for (int i = 0; i < ciclos; i++) {
        try {
            Thread.sleep(500);
        } catch (InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
        synchronized (this) {
            while (suspProc) {
                try{
                    wait();
                } catch (InterruptedException e)
                {
                    System.out.println(e.toString());
                }
            }

            if (paProc) break;
        }
    }
    estado = Estado.Terminado;
    modelo.setValueAt(estado, id, 1);
}
}

```

Resultado Final, ya una vez ejecutado el programa donde podemos ver la ejecución de las colas múltiples, siempre empezando por la de más alta prioridad:



Nombre	Estado	Prioridad	Uso de Memoria
System Idle Process	Terminado	Muy Alta	8
System	Espera	Muy Alta	20
Registry	Nuevo	Alta	27172
smss.exe	Nuevo	Baja	132
csrss.exe	Nuevo	Baja	1480
wininit.exe	Nuevo	Media	336
services.exe	Nuevo	Alta	6720
lsass.exe	Nuevo	Alta	14164
svchost9.exe	Nuevo	Media	396
svchost10.exe	Nuevo	Media	25792
fontdrvhost.exe	Nuevo	Alta	72
WUDFHost.exe	Nuevo	Alta	4756
svchost1.exe	Nuevo	Media	13700
svchost2.exe	Nuevo	Media	3424
svchost3.exe	Nuevo	Baja	4488
svchost4.exe	Ejecucion	Muy Alta	4068
svchost5.exe	Nuevo	Muy Alta	1612
svchost6.exe	Nuevo	Baja	1552
svchost7.exe	Nuevo	Baja	7464

Creditos

Conclusiones.

Con respecto a está actividad podemos entender lo practico que resulta el utilizar las colas múltiples en particular utilizándolas en un caso donde se explote todos los aspectos de lo que son capaces, como en el caso de usarlas para un caso donde se involucre la prioridad de los procesos, o en donde se tenga una clasificación para los mismos, en donde cada uno de ellos merezca de una atención muy particular o de un tiempo de ejecución distinto debido a las acciones que desempeñan en el sistema.

Con lo anterior podemos corroborar la importancia de este método y tener en cuenta los situaciones en que puede ser empleado, no solo para futuros desarrollos en cuanto a procesos, si no como método de mejora en futuras simulaciones con respecto a los procesos y los necesidades particulares de cada uno de ellos.

Referencias.

Colas Múltiples. (2009, 18 marzo). Recuperado 5 noviembre, 2019, de <https://karenescobarj.wordpress.com/2009/03/18/colas-multiples/>