Universidad Nacional del Altiplano

Educando mentes, Cambiando el mundo

Facultad de Ingeniería Mecánica Eléctrica, Electrónica y Sistemas

Escuela Profesional de Ingeniería de Sistemas

Simulación de estados de un proceso en C++

PARALELISMO, CONCURRENCIA Y SISTEMAS DISTRIBUIDOS

Ing. ROMERO FLORES ROBERT ANTONIO

estudiante

↬ **Larota Pilco David Brahyan** ↫

18 de abril de 2024

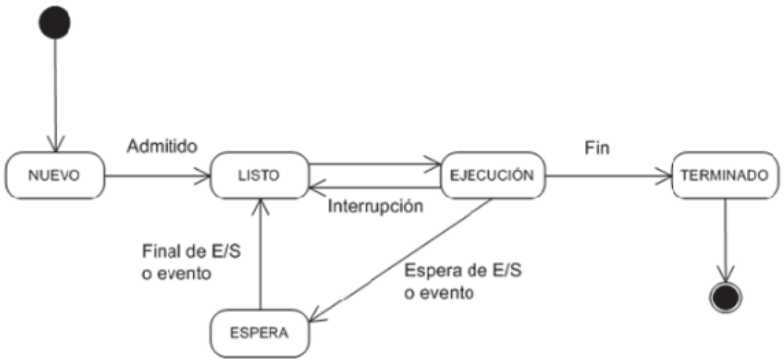
1 Introducción

La simulación de estados de un proceso es una técnica que permite estudiar el comportamiento de un proceso en diferentes situaciones. En el contexto de la programación en C++, esta simulación se puede realizar utilizando exclusión mutua e hilos. La exclusión mutua garantiza que solo un hilo pueda acceder a un recurso compartido a la vez, evitando posibles problemas de concurrencia. Por otro lado, los hilos permiten ejecutar diferentes tareas de forma concurrente, lo que resulta especialmente útil para simular los diferentes estados de un proceso. En esta sección, se explorará el concepto de estados de un proceso y la importancia de simularlos en C++.

2 Conceptos de Estados de un Proceso

Los estados de un proceso son las diferentes etapas por las que puede pasar un programa en su ejecución. Estos estados incluyen el estado de "listoçuando el proceso está preparado para ser ejecutado, el estado de .ejecuciónçuando el proceso está utilizando la CPU para ejecutar sus instrucciones, y el estado de "bloqueadoçuando el proceso está esperando la finalización de una operación de entrada/salida. Comprender estos estados es fundamental para entender cómo funciona un programa y cómo se comporta durante su ejecución. En esta sección, se ahondará en el concepto de los estados de un proceso y su relevancia en la simulación en C++.

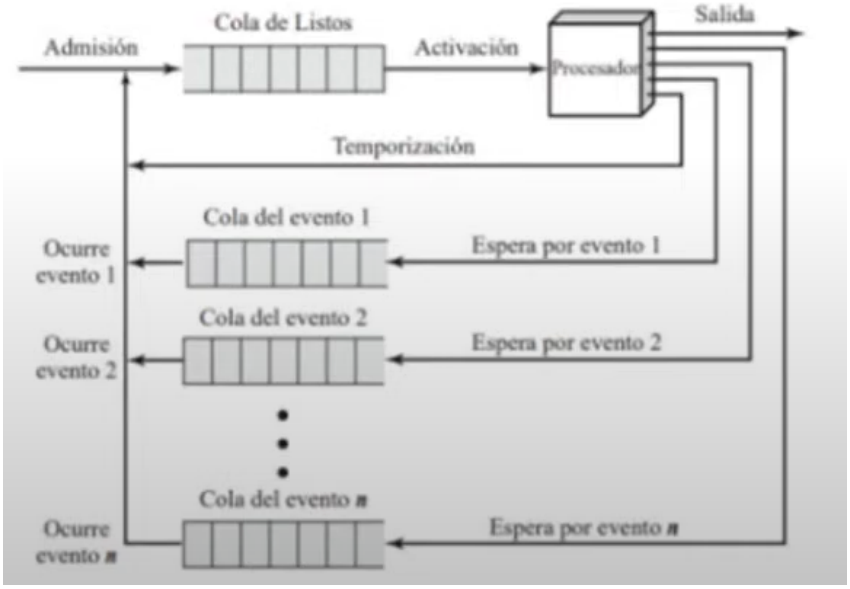
Figura 1: Estados de un proceso



3 Arquitectura de como Simular

Arquitectura del Modelo de Colas: Entonces tendremos una clase Proceso que tiene

Figura 2: Arquitectura del Modelo de Colas



los siguientes atributos

```
1
2  class Proceso
3  {
4  public:
5          int identificador;
6          estadoProceso estado;
7      Matriz contador;
8      Proceso() : identificador(ideProceso++), estado(NUEVO) ¬
          ↪ {}
9
10     Proceso(Matriz _cont) : identificador(ideProceso++), ¬
          ↪ estado(NUEVO), contador(_cont) {}
11
12 };
```

Siguiente cada proceso tiene instrucciones, las instrucciones para la simulación será una matriz de nx3. Donde n es la cantidad de instrucciones que tendrá el procesos, donde n = identificador de Introducción, la matriz[n,0] = identificador del recurso a utilizar por ejemplo: I/O como memoria, micrófono, mouse, parlantes, impresora, etc. por ultimo matriz[n,1] es el Tiempos de Ráfaga que el CPU le asigna a ese proceso, y por ultimo matriz[n,2] verifica si ese recurso ya ha sido ejecutado o no; si esta en 0 o falso quiere decir que no ha sido ejecutado si esta en 1 o true ya ha sido ejecutado.

```
1
```

```
2       enum estadoProceso
3   {
4       NUEVO,
5       LISTO,
6       EJECUCION,
7       BLOQUEADO,
8       TERMINADO
9   };
10
11  enum recursos
12  {
13      teclado,
14      camara,
15      audio,
16      mircrofono,
17      impresora,
18      disco_duro,
19      memoria
20  };
21
22  class Matriz {
23  private:
24
25  public:
26      int filas;
27      int columnas;
28      int** matriz;
29      Matriz(int filas = 1) : filas(filas), columnas(3) {
30          matriz = new int* [filas];
31
32          for (int i = 0; i < filas; ++i) {
33              matriz[i] = new int[columnas];
34              for (int j = 0; j < columnas; j++)
35              {
36                  matriz[i][j] = 0;
37              }
38          }
39      }
40
41      int obtenerValor(int fila, int columna) {
42          return matriz[fila][columna];
43      }
44      void establecerValor(int fila, int columna, int valor) {
45          matriz[fila][columna] = valor;
46      }
47      void setMatriz(int** nuevaMatriz) {
```

```
48          for (int i = 0; i < filas; ++i) {
49              for (int j = 0; j < columnas; ++j) {
50                  matriz[i][j] = nuevaMatriz[i][j];
51              }
52          }
53      }
54      void imprimirMatriz() {
55          for (int i = 0; i < filas; ++i) {
56              for (int j = 0; j < columnas; ++j) {
57                  std::cout << matriz[i][j] << " ";
58              }
59              std::cout << std::endl;
60          }
61      }
62  };
```

Luego procedemos a configurar el tiempo de simulación cada 1 segundo Asignamos valores:

```
1
2       void asignar()
3   {
4       a.establecerValor(0, 0, impresora); // recurso a usar
5       a.establecerValor(0, 1, 2);         // tiempos de rafaga ¬
          ↪ que nesecita
6
7       a.establecerValor(1, 0, teclado);
8       a.establecerValor(1, 1, 2);
9
10      a.establecerValor(2, 0, camara);
11      a.establecerValor(2, 1, 2);
12
13      a.establecerValor(3, 0, mircrofono);
14      a.establecerValor(3, 1, 2);
15
16      a.establecerValor(4, 0, impresora);
17      a.establecerValor(4, 1, 2);
18      pa.contador = a; pa.estado = LISTO; sim.listos.push_back¬
          ↪ (pa);
19
20
21
22      b.establecerValor(0, 0, teclado); // recurso a usar
23      b.establecerValor(0, 1, 20);        // tiempos de rafaga¬
          ↪  que nesecita
24
25      b.establecerValor(1, 0, camara);
26      b.establecerValor(1, 1, 50);
```

```
27
28      b.establecerValor(2, 0, mircrofono);
29      b.establecerValor(2, 1, 60);
30
31      b.establecerValor(3, 0, disco_duro);
32      b.establecerValor(3, 1, 80);
33
34      b.establecerValor(4, 0, memoria);
35      b.establecerValor(4, 1, 40);
36      pb.contador = b; pb.estado = LISTO; sim.listos.push_back
          ↪ (pb);
37
38
39      c.establecerValor(0, 0, memoria); // recurso a usar
40      c.establecerValor(0, 1, 20);        // tiempos de rafaga
          ↪  que nesecita
41
42      c.establecerValor(1, 0, teclado);
43      c.establecerValor(1, 1, 80);
44
45      c.establecerValor(2, 0, camara);
46      c.establecerValor(2, 1, 60);
47
48      c.establecerValor(3, 0, disco_duro);
49      c.establecerValor(3, 1, 20);
50
51      c.establecerValor(4, 0, teclado);
52      c.establecerValor(4, 1, 10);
53      pc.contador = c; pc.estado = LISTO; sim.listos.push_back
          ↪ (pc);
54
55
56      d.establecerValor(0, 0, audio); // recurso a usar
57      d.establecerValor(0, 1, 20);        // tiempos de rafaga
          ↪  que nesecita
58
59      d.establecerValor(1, 0, teclado);
60      d.establecerValor(1, 1, 10);
61
62      d.establecerValor(2, 0, camara);
63      d.establecerValor(2, 1, 60);
64
65      d.establecerValor(3, 0, mircrofono);
66      d.establecerValor(3, 1, 50);
67
68      d.establecerValor(4, 0, impresora);
```

```
69      d.establecerValor(4, 1, 10);
70      pd.contador = d; pd.estado = LISTO; sim.listos.push_back
           ↪ (pd);
71
72
73      e.establecerValor(0, 0, disco_duro); // recurso a usar
74      e.establecerValor(0, 1, 20);         // tiempos de rafaga
           ↪  que nesecita
75
76      e.establecerValor(1, 0, memoria);
77      e.establecerValor(1, 1, 40);
78
79      e.establecerValor(2, 0, camara);
80      e.establecerValor(2, 1, 60);
81
82      e.establecerValor(3, 0, mircrofono);
83      e.establecerValor(3, 1, 10);
84
85      e.establecerValor(4, 0, audio);
86      e.establecerValor(4, 1, 10);
87      pe.contador = e; pe.estado = LISTO; sim.listos.push_back
           ↪ (pe);
88      /*for (auto i = 0; i < sim.listos.size(); i++)
89      {
90          cout << sim.listos[i].identificador << endl;
91      }*/
92  }
```

(Autor, 2022)

4 Referencias

Referencias

Autor. (2022). prueba del documento lenguaje latex. *https://www.overleaf.com/*, *13*(36), 34–36.