Instituto Politécnico Nacional

Escuela Superior de Cómputo

Tarea 2

Unidad de aprendizaje: Sistemas operativos

Grupo: 2CM8

Alumnos:

- Martínez Coronel Brayan Yosafat
- Monteros Cervantes Miguel Angel
- Ramírez Olvera Guillermo
- Sánchez Méndez Edmundo Josué

Profesor:
Cortés Galicia Jorge

Algorithm 1: Sustitución de páginas FIFO

Entrada: Número de Marcos numM, Número de paginas numP, Cola de paginas P
Salida: Número de fallos numF
Funcion: BuscaFalloPagina($numM, numP, P$)

```
/* Supongamos creada una estructura de datos tipo cola                    */
1  Marco = nuevaCola();
2  numF = 0;
3  for k = 0 hasta numP do
4      if Marco.length < numM then
5          if P[k] no esta en Marco then
6              Marco.formar(Marco,P[k]);
7              numF = numF + 1;
8      else
9          if P[k] no esta en Marco then
10             Marco.desformar(Marco);
11             Marco.formar(Marco,P[k]);
12             numF = numF + 1;
13 return numF;
```

Algorithm 2: Sustitución óptima de páginas

Entrada: Número de Marcos numM, Número de paginas numP, Arreglo de paginas P
Salida: Número de fallos numF

```
/* Se usaran dos funciones, la que nos dara el numero de fallos y la que buscara la pagina con el mayor
   periodo de tiempo en no ser utilizada                                                              */
```

Funcion: Prediccion$P, Marco, numP, inicioPrediccion$)

```
1  prediccion = −1;
2  masLejano = inicioPrediccion;
3  for i = 0 hasta Marco.length do
4      for j = inicioPrediccion hasta numP do
5          if Marco[i] == P[j] then
6              if j > masLejano then
7                  masLejano = j;
8                  prediccion = i;
9              break;
10     if j == numP then
11         return i;
12 return (prediccion == -1) ? 0 : prediccion;
```

Funcion: BuscaFalloPagina($numM, numP, P$)

```
13 Marco = nuevoVector();
14 numF = 0;
15 for k = 0 hasta numP do
16     if P[k] esta en Marco then
17         continuar;
18     numF=numF+1;                                  // Si no esta, entonces hay fallo
19     if Marco.length < numM then
20         S.aniadirFinal(P[k]);                     // Añade al final del vector el valor
21     else
22         indicePrediccion = Prediccion(P, Marco, numP, i + 1);
23         Marco[indicePrediccion]=P[k];
24 return numF;
```

Algorithm 3: Sustitución de páginas LRU

Entrada: Número de Marcos numM, Arreglo de paginas P
Salida: Número de fallos numF
/* Recordar que podemos usar tanto contadores como pilas, asi que el desarrollo de este algoritmo es la idea general */
Funcion: BuscaFalloPagina($numM, P$)
/* Supongamos una estructura de datos tipo lista creada */

```
1  Marco = vacia();
2  numF = 0;
3  auxiliar = 0;
4  foreach i en P do
5      if !Marco.Busca(i) then
6          if Marco.length == numM then
7              Marco.Eliminar(0);
8              Marco.Añadir(numM-1,i);          // Indice, Número a introducir
9          else
10             Marco.Añadir(auxiliar,i);
11         numF=numF+1;
12         auxiliar=auxiliar+1;
13     else
14         Marco.Eliminar(i);                   // Indice, Número a introducir
15         Marco.add(Marco.length,i);           // Indice, Número a introducir
16 return numF;
```

Algorithm 4: Sustitución de páginas mediante aproximación LRU "Bits de referencia adicionales"

Entrada: Número de Marcos numM, Número de paginas numP, Cola de paginas P
Salida: Número de fallos numF
Funcion: BuscaFalloPagina($numM, numP, P$)
/* Supongamos creada una estructura de datos tipo cola */

```
1  Marco = nuevaCola();
2  numF = 0;
3  for k = 0 hasta numP do
4      if Marco.length < numM then
5          if P[k] no esta en Marco then
6              Marco.formar(Marco,P[k]);
7              numF = numF + 1;
8      else
9          if P[k] no esta en Marco then
10             Marco.desformar(Marco);
11             Marco.formar(Marco,P[k]);
12             numF = numF + 1;
13 return numF;
```