Page replacement Algorithm

```
kaushik@kaushik-AcerPower-Series:~/OS $ cat pagerepl.c
#include<stdio.h>
#include<string.h>
#define NO 0
#define YES 1
#define REPLACED 2
#define FOUND 1
#define NOTFOUND 0
struct Frames
{
int num, age, marker;
};
void ReadInput(char *refString, int *FrameNum, int *FreeFrames);
void FIFO(struct Frames *f, char *refString, int *FrameNum, int len);
void LRU(struct Frames *f, char *refString, int *FrameNum, int len);
void Optimal(struct Frames *f, char *refString, int *FrameNum, int
len);
void display(char Page, struct Frames *f, int FrameNum, int PF);
void initialize(struct Frames *f, int FrameNum);
int main()
{
struct Frames *f;
int FrameNum=0, FreeFrames=0, len;
int i=0, j=0, k=0, Choice=0;
char refString[100];
while(Choice<5)
{
printf("\nPAGE REPLACEMENT ALGORITHMS\n\t1.Read
Input\n\t2.FIFO\n\t3.LRU\n\t4.Optimal\n\t5.Exit\nEnter Choice: ");
scanf("%d", &Choice);
switch(Choice)
{
case 1:
ReadInput(refString, &FrameNum, &FreeFrames);
f = malloc(FrameNum * sizeof(struct Frames));
initialize(f, FrameNum);
len = strlen(refString);
printf("\n");
break;
case 2:
printf("\n\tFIFO PAGE REPLACEMENT ALGORITHM\nThe
reference string is %s\n\n", refString);
printf("Page\tFrames\tPageFault\n");
FIFO(f, refString, &FrameNum, len);
initialize(f, FrameNum);
break;
case 3:
printf("\n\tLRU PAGE REPLACEMENT ALGORITHM\nThe
reference string is %s\n\n", refString);
printf("Page\tFrames\tPageFault\n");
```

```
LRU(f, refString, &FrameNum, len);
initialize(f, FrameNum);
break;
case 4:
printf("\n\tOPTIMAL PAGE REPLACEMENT
ALGORITHM\nThe reference string is %s\n\n", refString);
printf("Page\tFrames\tPageFault\n");
Optimal(f, refString, &FrameNum, len);
initialize(f, FrameNum);
break;
case 5:
default:
break;
}
}
}
void ReadInput(char *refString, int *FrameNum, int *FreeFrames)
{
printf("\nEnter number of Free Frames: ");
scanf("%d", FreeFrames);
printf("\nEnter number of frames required by the process: ");
scanf("%d", FrameNum);
printf("\nEnter the reference string: ");
scanf("%s", refString);
}
void FIFO(struct Frames *f, char *refString, int *FrameNum, int len)
{
int count=0, pageFaults=0, flag, minAge, PF=0;
int i=0, k=0, j=0, x=0, y=0;
while(k<len)
{
while(count!=*FrameNum&&k<len)
{
flag = NOTFOUND;
for(j=0;j< *FrameNum; j++)
if(f[j].num == refString[k]48&&
f[j].age!=0)
{
flag = FOUND;
PF=0;
break;
}
if(flag == NOTFOUND)
{
pageFaults++;
minAge = f[0].age;
i=0;
for(j=1; j< *FrameNum; j++)
if(f[j].age<minAge)
{
minAge=f[j].age;
```

```
i=j;
}
if(i<*FrameNum&&k!=0)
{
for(j=i+1;j< *FrameNum;j++)
if(f[j].num == 1)
{
i=j;
break;
}
}
flag = REPLACED;
f[i].num = refString[k]48;
f[i].age = k;
if(flag == REPLACED)
PF=1;
}
display(refString[k], f, *FrameNum, PF);
count++;
k++;
}
count = 0;
}
printf("\nTotal number of Page Faults = %d\n", pageFaults);
}
void LRU(struct Frames *f, char *refString, int *FrameNum, int len)
{
int count=0, pageFaults=0, flag, maxAge, PF=0;
int i=0, k=0, j=0, x=0, y=0, p=0, q=0, tempCount=0;
char enter;
while(k<len)
{
while(count<*FrameNum&&k<len)
{
flag = NOTFOUND;
for(j=0;j<*FrameNum;j++)
{
if(f[j].num==1)
{
pageFaults++;
i=j;
flag= REPLACED;
break;
}
else
if(f[j].num==refString[k]48)
{
i=j;
flag=FOUND;
PF=0;
break;
```

```
}
}
if(flag==NOTFOUND)
{
pageFaults++;
tempCount=0;
p=1;
while(tempCount< *FrameNum1&&
p<len)
{
for(j=0;j<*FrameNum;j++)
{
if(refString[kp]
48==
f[j].num)
{
if(f[j].marker!=YES)
{
f[j].marker=YES;
tempCount++;
}
break;
}
}
if(p<len)
p++;
}
for(j=0;j<*FrameNum;j++)
if(f[j].marker==NO)
{
i=j;
break;
}
f[i].age=0;
flag = REPLACED;
}
f[i].num = refString[k]48;
if(flag == REPLACED)
PF = 1;
display(refString[k], f, *FrameNum, PF);
count++;
k++;
for(j=0;j<*FrameNum;j++)
{
f[j].marker=NO;
if(f[j].num!=1)
f[j].age++;
}
}
count = 0;
}
```

```
printf("\nTotal number of Page Faults = %d\n", pageFaults);
}
void Optimal(struct Frames *f, char *refString, int *FrameNum, int
len)
{
int count=0, pageFaults=0, flag, maxAge, PF=0;
int i=0, k=0, j=0, x=0, y=0, p=0, q=0, tempCount=0;
while(k<len)
{
while(count<*FrameNum&&k<len)
{
flag = NOTFOUND;
for(j=0;j<*FrameNum;j++)
{
if(f[j].num==1)
{
pageFaults++;
i=j;
flag= REPLACED;
break;
}
else
if(f[j].num==refString[k]48)
{
i=j;
flag=FOUND;
PF=0;
break;
}
}
if(flag==NOTFOUND)
{
pageFaults++;
tempCount=0;
p=1;
while(tempCount< *FrameNum1&&
p<len)
{
for(j=0;j<*FrameNum;j++)
{
if(refString[k+p]48==
f[j].num)
{
if(f[j].marker!=YES)
{
f[j].marker=YES;
tempCount++;
}
break;
}
}
```

```
if(p<len)
p++;
}
for(j=0;j<*FrameNum;j++)
if(f[j].marker==NO)
{
i=j;
break;
}
f[i].age=0;
flag = REPLACED;
}
f[i].num = refString[k]48;
if(flag == REPLACED)
PF=1;
display(refString[k], f, *FrameNum, PF);
count++;
k++;
for(j=0;j<*FrameNum;j++)
{
f[j].marker=NO;
if(f[j].num!=1)
f[j].age++;
}
}
count = 0;
}
printf("\nTotal number of Page Faults = %d\n", pageFaults);
}
void display(char Page, struct Frames *f, int FrameNum, int PF)
{
int i;
printf("\n%c\t", Page);
for(i=0;i<FrameNum;i++)
{
if(PF==1)
if(f[i].num!=1)
printf("%d ",f[i].num);
else
printf(" ");
else
printf(" ");
}
printf("\t%d", PF);
}
void initialize(struct Frames *f, int FrameNum)
{
int i;
for(i=0;i<FrameNum;i++)
{
f[i].num=1;
```

```
f[i].age=0;
f[i].marker=NO;
}
}
kaushik@kaushik-AcerPower-Series:~/OS $ gcc pagerepl.c
kaushik@kaushik-AcerPower-Series:~/OS $ ./a.out
PAGE REPLACEMENT ALGORITHMS
1.Read Input
2.FIFO
3.LRU
4.Optimal
5.Exit
Enter Choice: 1
Enter number of Free Frames: 10
Enter number of frames required by the process: 3
Enter the reference string: 70120304230321201701
PAGE REPLACEMENT ALGORITHMS
1.Read Input
2.FIFO
3.LRU
4.Optimal
5.Exit
Enter Choice: 2
FIFO PAGE REPLACEMENT ALGORITHM
The reference string is 70120304230321201701
Page Frames PageFault
7 7 1
0 7 0 1
1 7 0 1 1
2 2 0 1 1
0 0
3 2 3 1 1
0 2 3 0 1
4 4 3 0 1
2 4 2 0 1
3 4 2 3 1
0 0 2 3 1
3 0
2 0
1 0 1 3 1
2 0 1 2 1
0 0
1 0
7 7 1 2 1
0 7 0 2 1
1 7 0 1 1
Total number of Page Faults = 15
PAGE REPLACEMENT ALGORITHMS
1.Read Input
2.FIFO
3.LRU
```

4.Optimal
5.Exit
Enter Choice: 3
LRU PAGE REPLACEMENT ALGORITHM
The reference string is 70120304230321201701
Page Frames PageFault
7 7 1
0 7 0 1
1 7 0 1 1
2 2 0 1 1
0 0
3 2 0 3 1
0 0
4 4 0 3 1
2 4 0 2 1
3 4 3 2 1
0 0 3 2 1
3 0
2 0
1 1 3 2 1
2 0
0 1 0 2 1
1 0
7 1 0 7 1
0 0
1 0
Total number of Page Faults = 12
PAGE REPLACEMENT ALGORITHMS
1.Read Input
2.FIFO
3.LRU
4.Optimal
5.Exit
Enter Choice: 4
OPTIMAL PAGE REPLACEMENT ALGORITHM
The reference string is 70120304230321201701
Page Frames PageFault
7 7 1
0 7 0 1
1 7 0 1 1
2 2 0 1 1
0 0
3 2 0 3 1
0 0
4 2 4 3 1
2 0
3 0
0 2 0 3 1
3 0
2 0
1 2 0 1 1

2 0
0 0
1 0
7 7 0 1 1
0 0
1 0
Total number of Page Faults = 9
PAGE REPLACEMENT ALGORITHMS
1.Read Input
2.FIFO
3.LRU
4.Optimal
5.Exit
Enter Choice: 5