Gebze Technical University
Computer Engineering

CSE 312 - 2020 Spring

FINAL REPORT

TAYLAN ÖNDER
151044015

1 INTRODUCTION

1.1 Problem Definition

Virtual memory management system and a number of page replacement algorithms implemented this Project.4 different sorting algorithms (Bubble sort, quick sort, merge sort, index sort) with threads are requested to simulate the normal working principle with the help of physical and virtual memory.

2 METHOD

2.1 Problem Solution Approach

PART 1

Page Frame Number	Modified	Referenced	Absent	Type	Count	PriorityNum

According to Fig 3-11 in my own alogirithm my typical page table entry is like that.Page frame number,modified and referenced attributes also included in table 3-11.Type,count and priorityNum variables for pagereplacement algorithm helper attributes. In order to explain them;

Page Frame Number : This is an index to indicate physical memory in the elements (integers) used.

```
if(virtualAddressSpace[boxIndex].absentorNot==1){
   virtualAddressSpace[boxIndex].modified = 1;
   virtualAddressSpace[boxIndex].referenced = 1;
   if(strcmp(pageReplacement,"LRU")==0){
      for(i=0;i<numVirtual;i++){
          if(virtualAddressSpace[i].absentorNot==1 ){
              virtualAddressSpace[i].usedCounter--;
          }
      }
      virtualAddressSpace[boxIndex].usedCounter=0;
   }
   virtualAddressSpace[boxIndex].type='q';
   pthread_mutex_unlock(&lock);
   return physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize];
}
```

For including in frame of physical memory, with index(page frame number) is Access way to using physical memory elements (integers).

Modified : Indicates whether the frame loaded on the physical memory is used with the get or set operation.I used for writing disk data, check modified or not.If frame is not modified, I didn't change anything in disk for frame.Another usage, I check in NRU page replacement algortihm for 4 type relation referenced and modified.

Referenced : When frame access is provided with get and set, I changed it to reference 1. I used the replacements algorithms for SC and NRU algorithms.

```
659  void nruMethod(int boxIndex){
660      int i=0,control=0,NRUindex=-1;
661      for(i=0;i<numVirtual;i++){
662          if(control==0){
663              if(virtualAddressSpace[i].absentorNot==1 && virtualAddressSpace[i].modified==0 && virtualAddressSpace[i].referenced==0){
664                  NRUindex=i;
665                  break;
666              }
667              if(i==numVirtual-1){
668                  i=0;
669                  control=1;
670              }
671          }
```

Example of modified and referenced using in NRU algorithm for replace frame.

Absent : It represent the searching data is in physical memory or not.To read disk frames firstly check in page frame this index is in physical memory or not.All accessing physical memory process firstly checked absent or not. When replacement algorithm working I change absent -2 for deleting in physical memory and writing disk file.

```
728          else if(strcmp(tName,"quick")==0){
729              printStatistic[1].numberOfReads++;
730               boxIndex=index/frameSize;
731               if(virtualAddressSpace[boxIndex].absentorNot==1){
732                  virtualAddressSpace[boxIndex].modified = 1;
733                  virtualAddressSpace[boxIndex].referenced = 1;
734                  if(strcmp(pageReplacement,"LRU")==0){
735                     for(i=0;i<numVirtual;i++){
736                         if(virtualAddressSpace[i].absentorNot==1 ){
737                             virtualAddressSpace[i].usedCounter--;
738                         }
739                     }
740                     virtualAddressSpace[boxIndex].usedCounter=0;
741                  }
742                  virtualAddressSpace[boxIndex].type='q';
743                  pthread_mutex_unlock(&lock);
744                  return physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize];
745              }
746              else{
747                  printStatistic[1].numberOfPageMisses++;
748                  if(indexAdressPyhsical>=numPhysical){
749                      if(strcmp(pageReplacement,"LRU")==0){
750                          lruMethod(boxIndex);
751                          printStatistic[1].numberOfPageReplacements++;
752                          printStatistic[1].diskPageWrite++;
753                      }
754                      else if(strcmp(pageReplacement,"NRU")==0){
755                          nruMethod(boxIndex);
756                          printStatistic[1].numberOfPageReplacements++;
757                          printStatistic[1].diskPageWrite++;
758                      }
```

Type : Type is used for when finish the sorting algorithm, It is used to write the last remaining data to the disk in physical memory. Since there are programs running at the same time, I know which sorts the frame in the pageframes belong to and write them to the file in that sort.

```
155  void* mergeFunction(void *a){
156      pthread_mutex_lock(&sequential);
157      int i=0,j=0,valueTemp=0;
158      mergeSortFunct(2*numLength,(3*numLength)-1);
159      for(j=0;j<numVirtual;j++){
160          if(virtualAddressSpace[j].absentorNot==1 && virtualAddressSpace[j].type=='m'){
161              pthread_mutex_lock(&lock);
162              int indexParam=j;
163              int i=0,value=0;
164              int physicalIndex=virtualAddressSpace[j].index;
165              int startAdress=virtualAddressSpace[j].startAdress;
166              printStatistic[3].diskPageWrite++;
167              fseek(diskFile,startAdress*sizeof(int),SEEK_SET);
168              for(i=0;i<frameSize;i++){
169                  value=physicalframes[physicalIndex][i];
170                  fwrite(&value , sizeof(int) , 1 , diskFile );
171              }
172              virtualAddressSpace[j].absentorNot=-2;
173              pthread_mutex_unlock(&lock);
174          }
175      }
176      pthread_mutex_unlock(&sequential);
177
178  }
179
```

Example of merge sort,check with absent and type field for sort algorithms (change according to agortihm type) remaining data in physical memory need to be write disk or not.

Count :It is for just a LRU algorithm.For accessing get and set methods I modified this field zero for index of pageframe and I decrease this value of other frames by one. my main goal here is to find the most recently used (outdated) frame and write that frame to disk and fill in the frame that I need instead of in physical memory.

```
645  void lruMethod(int boxIndex){
646      int LRUindex=0,LRUindexValue=0,i=0;
647      LRUindexValue=numVirtual*frameSize;
648      for(i=0;i<numVirtual;i++){
649          if(virtualAddressSpace[i].absentorNot==1 && LRUindexValue>virtualAddressSpace[i].usedCounter){
650              LRUindexValue=virtualAddressSpace[i].usedCounter;
651              LRUindex=i;
652          }
653      }
654      writeFile(LRUindex);
655      virtualAddressSpace[LRUindex].absentorNot=-2;
656      virtualAddressSpace[boxIndex].index=virtualAddressSpace[LRUindex].index;
657  }
658
```

According to value, find replacement index for LRU algorithm.

PriorityNum: It used for SC and FIFO algorithm to used order of save frame in physical memory.According to this field, find the index for replacement index.

```
struct virtualAddress {
    int index;
    int referenced;
    int modified;
    int absentorNot;
    int usedCounter;
    int startAdress;
    int priorityNum;
    char type;
};
virtualAddress *virtualAddressSpace;
```

View of the page table in the code (Line 47)

PART 2:

I begin with creating thread and filling with fill thread in data file random numbers.

```
372  void* fillRandomNumber(void* frameSizeInput){
373      pthread_mutex_lock(&sequential);
374      srand(1000);
375
376      int i=0,j=0,index=0;
377      char type[10]="fill";
378      for (i = 0; i < numVirtual; i++)
379      {
380          for (j = 0; j < frameSize; j++)
381          {
382              set(index, (int) rand(),type );
383              index++;
384          }
385      }
386      sem_post(&mutex);
387      pthread_mutex_unlock(&sequential);
388
389      pthread_exit(NULL);
390
391  }
392
```

This thread method create random numbers and with set method write datas in file.Type fill in set write in a file.I also use mutex for other threads can not start without before writing to the file is complete disk file.

```
529  void set(unsigned int index, int value, char * tName)
530  {
531      pthread_mutex_lock(&lock);
532      int i=0;
533      pageTableCounter++;
534      if(pageTableCounter!=0 && (pageTableCounter%pageTablePrintInt)==0){
535          for(i=0;i<numVirtual;i++){
536              printf("Page table frame %d \n",i );
537              printf("Page table index %d \n",virtualAddressSpace[i].index );
538              printf("Page table referenced %d \n",virtualAddressSpace[i].referenced );
539              printf("Page table modified %d \n",virtualAddressSpace[i].modified );
540              printf("Page table absent %d \n\n",virtualAddressSpace[i].absentorNot );
541          }
542          printf("\n\n");
543      }
544      int indexAddress=0;
545      int boxIndex=index/frameSize;
546      if(strcmp(tName,"fill")==0){
547          printStatistic[0].numberOfWrites++;
548          printStatistic[0].diskPageWrite++;
549          fwrite(&value ,  sizeof(int),1  , diskFile );
550
```

```
529 void set(unsigned int index, int value, char * tName)
530 {
531     pthread_mutex_lock(&lock);
532     int i=0;
533     pageTableCounter++;
534     if(pageTableCounter!=0 && (pageTableCounter%pageTablePrintInt)==0){
535         for(i=0;i<numVirtual;i++){
536             printf("Page table frame %d \n",i );
537             printf("Page table index %d \n",virtualAddressSpace[i].index );
538             printf("Page table referenced %d \n",virtualAddressSpace[i].referenced );
539             printf("Page table modified %d \n",virtualAddressSpace[i].modified );
540             printf("Page table absent %d \n\n",virtualAddressSpace[i].absentorNot );
541         }
542         printf("\n\n");
543     }
544     int indexAddress=0;
545     int boxIndex=index/frameSize;
546     if(strcmp(tName,"fill")==0){
547         printStatistic[0].numberOfWrites++;
548         printStatistic[0].diskPageWrite++;
549         fwrite(&value ,  sizeof(int),1   , diskFile );
550 
551     }
552     else if(strcmp(tName,"quick")==0){
553         virtualAddressSpace[boxIndex].modified = 1;
554         virtualAddressSpace[boxIndex].referenced = 1;
555         virtualAddressSpace[boxIndex].type='q';
556         printStatistic[1].numberOfWrites++;
557         physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize]=value;
558     }
559     else if(strcmp(tName,"bubble")==0){
560         virtualAddressSpace[boxIndex].modified = 1;
561         virtualAddressSpace[boxIndex].referenced = 1;
562         virtualAddressSpace[boxIndex].type='b';
563         printStatistic[2].numberOfWrites++;
564         physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize]=value;
565 
566     }
567     else if(strcmp(tName,"merge")==0){
568         virtualAddressSpace[boxIndex].type=='m';
569         virtualAddressSpace[boxIndex].modified = 1;
570         virtualAddressSpace[boxIndex].referenced = 1;
571         printStatistic[3].numberOfWrites++;
572         physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize]=value;
573 
574     }
```

Get-set methods provides Access pjysical memory datas to set and get.According to thread type, I keep the ststistical values for printing finish of program.Modified,reference and the othor all page table fields change and in this two main method.And also print according to value in giving input parameter, I keep a counter and write pagetable fields.

```
int get(unsigned int index, char * tName)
{
    pthread_mutex_lock(&lock);
    int indexAddress=0,value=0,i=0,j=0;
    int boxIndex=0;
    int valueTemp=0,control=0;
    pageTableCounter++;
    if(pageTableCounter!=0 && (pageTableCounter%pageTablePrintInt)==0){
        for(i=0;i<numVirtual;i++){
            printf("Page table frame %d \n",i );
            printf("Page table index %d \n",virtualAddressSpace[i].index );
            printf("Page table referenced %d \n",virtualAddressSpace[i].referenced );
            printf("Page table modified %d \n",virtualAddressSpace[i].modified );
            printf("Page table absent %d \n\n",virtualAddressSpace[i].absentorNot );
        }
        printf("\n\n");
    }
    if(strcmp(tName,"fill")==0){
        fread(&value, sizeof(int),1, diskFile);
    }
    else if(strcmp(tName,"quick")==0){
        printStatistic[1].numberOfReads++;
         boxIndex=index/frameSize;
         if(virtualAddressSpace[boxIndex].absentorNot==1){
            virtualAddressSpace[boxIndex].modified = 1;
            virtualAddressSpace[boxIndex].referenced = 1;
            if(strcmp(pageReplacement,"LRU")==0){
               for(i=0;i<numVirtual;i++){
                    if(virtualAddressSpace[i].absentorNot==1 ){
                        virtualAddressSpace[i].usedCounter--;
                    }
                }
                virtualAddressSpace[boxIndex].usedCounter=0;
            }
            virtualAddressSpace[boxIndex].type='q';
            pthread_mutex_unlock(&lock);
            return physicalframes[virtualAddressSpace[boxIndex].index][index%frameSize];
        }
        else{
            printStatistic[1].numberOfPageMisses++;
            if(indexAdressPyhsical>=numPhysical){
                if(strcmp(pageReplacement,"LRU")==0){
                    lruMethod(boxIndex);
                    printStatistic[1].numberOfPageReplacements++;
                    printStatistic[1].diskPageWrite++;
                }
                else if(strcmp(pageReplacement,"NRU")==0){
                    nruMethod(boxIndex);
                    printStatistic[1].numberOfPageReplacements++;
                    printStatistic[1].diskPageWrite++;
                }
                else if(strcmp(pageReplacement,"FIFO")==0){
                    fifoMethod(boxIndex);
                    printStatistic[1].numberOfPageReplacements++;
                    printStatistic[1].diskPageWrite++;
                }
                else if(strcmp(pageReplacement,"SC")==0){
                    scMethod(boxIndex);
                    printStatistic[1].numberOfPageReplacements++;
                    printStatistic[1].diskPageWrite++;
                }
```

Get is maybe the most important method in program.It provides keeping and seting pagetable and physical memory fields.Print page frame or run page replacement algorithms or geting datas with frames in disk file.For senchronization, I used lock mutex to only one thread can Access when changing or getting or reading and writing input file (when replacing algorithm working).

Page Replacement Algorithims

FIFO:

```
610  void fifoMethod(int boxIndex){
611      int i=0,FIFOindex=0,FIFOindexValue=0;
612      FIFOindexValue=numVirtual*frameSize;
613      for(i=0;i<numVirtual;i++){
614          if(virtualAddressSpace[i].absentorNot==1){
615              if(FIFOindexValue>virtualAddressSpace[i].priorityNum){
616                  FIFOindexValue=virtualAddressSpace[i].priorityNum;
617                  FIFOindex=i;
618              }
619          }
620      }
621      writeFile(FIFOindex);
622      virtualAddressSpace[FIFOindex].absentorNot=-2;
623      virtualAddressSpace[boxIndex].index=virtualAddressSpace[FIFOindex].index;
624  }
625
```

FIFO is like first in first out prenciple. In this algorithm, the purpose is to find the frame that was saved memory first to find the frame to be written instead of.According to priorityNum pagetable field, find the oldest saved frame in memory.

SC

```
569  void scMethod(int boxIndex){
570      int i=0,SCindexValue=0,SCindex=-1,flag=0;
571      SCindexValue=numVirtual*frameSize;
572      for(i=0;i<numVirtual;i++){
573          if(virtualAddressSpace[i].absentorNot==1){
574              if(SCindexValue>virtualAddressSpace[i].priorityNum){
575                  if(virtualAddressSpace[i].referenced==1){
576                      virtualAddressSpace[i].referenced=0;
577                  }
578                  else{
579                      SCindexValue=virtualAddressSpace[i].priorityNum;
580                      SCindex=i;
581                      if(flag==1)
582                      break;
583                  }
584
585              }
586          }
587          if(i==numVirtual-1 && SCindex==-1){
588              flag=1;
589              i=0;
590              for(i=0;i<numVirtual;i++){
591                  if(virtualAddressSpace[i].absentorNot==1){
592                      SCindexValue=virtualAddressSpace[i].priorityNum;
593                      SCindex=i;
594                      break;
595
596                  }
597              }
598              break;
599          }
600      }
601      writeFile(SCindex);
602      virtualAddressSpace[SCindex].absentorNot=-2;
603      virtualAddressSpace[boxIndex].index=virtualAddressSpace[SCindex].index;
604  }
605
```

Very similar to FIFO algorithm. the only difference is that when finding the index, it is also intended to be 0 in the representation field in the page table.1 replacement fields are not taken, but the replacement fields are set to 0. If the field is 1, the frame in the physical

memory, which is in the first ready position from the indexes in the physical page table, according to the index in the pagetable when the loop ends for all frames. index is selected.

LRU

```
void lruMethod(int boxIndex){
    int LRUindex=0,LRUindexValue=0,i=0;
    LRUindexValue=numVirtual*frameSize;
    for(i=0;i<numVirtual;i++){
        if(virtualAddressSpace[i].absentorNot==1 && LRUindexValue>virtualAddressSpace[i].usedCounter){
            LRUindexValue=virtualAddressSpace[i].usedCounter;
            LRUindex=i;
        }
    }
    writeFile(LRUindex);
    virtualAddressSpace[LRUindex].absentorNot=-2;
    virtualAddressSpace[boxIndex].index=virtualAddressSpace[LRUindex].index;
}
```

Set and get operation is aimed to find the oldest frame.Pageframe field of UsedCounter used for comparing.When get or set working on any frame, all frame usedcounter values decrease and seting frame value set 0.0 is meaning of new used.

NRU

```
void nruMethod(int boxIndex){
    int i=0,control=0,NRUindex=-1;
    for(i=0;i<numVirtual;i++){
        if(control==0){
            if(virtualAddressSpace[i].absentorNot==1 && virtualAddressSpace[i].modified==0 && virtualAddressSpace[i].referenced==0){
                NRUindex=i;
                break;
            }
            if(i==numVirtual-1){
                i=0;
                control=1;
            }
        }
        else if(control==1){
            if(virtualAddressSpace[i].absentorNot==1 && virtualAddressSpace[i].modified==1 && virtualAddressSpace[i].referenced==0){
                NRUindex=i;
                break;
            }
            if(i==numVirtual-1){
                i=0;
                control=2;
            }
        }
        else if(control==2){
            if(virtualAddressSpace[i].absentorNot==1 && virtualAddressSpace[i].modified==0 && virtualAddressSpace[i].referenced==1){
                NRUindex=i;
                break;
            }
            if(i==numVirtual-1){
                i=0;
                control=3;
            }
        }
        else if(control==3){
            if(virtualAddressSpace[i].absentorNot==1 && virtualAddressSpace[i].modified==1 && virtualAddressSpace[i].referenced==1){
                NRUindex=i;
                break;
            }
            if(i==numVirtual-1){
                i=0;
                break;
            }
        }
    }
    writeFile(NRUindex);
    virtualAddressSpace[NRUindex].absentorNot=-2;
    virtualAddressSpace[boxIndex].index=virtualAddressSpace[NRUindex].index;
}

```

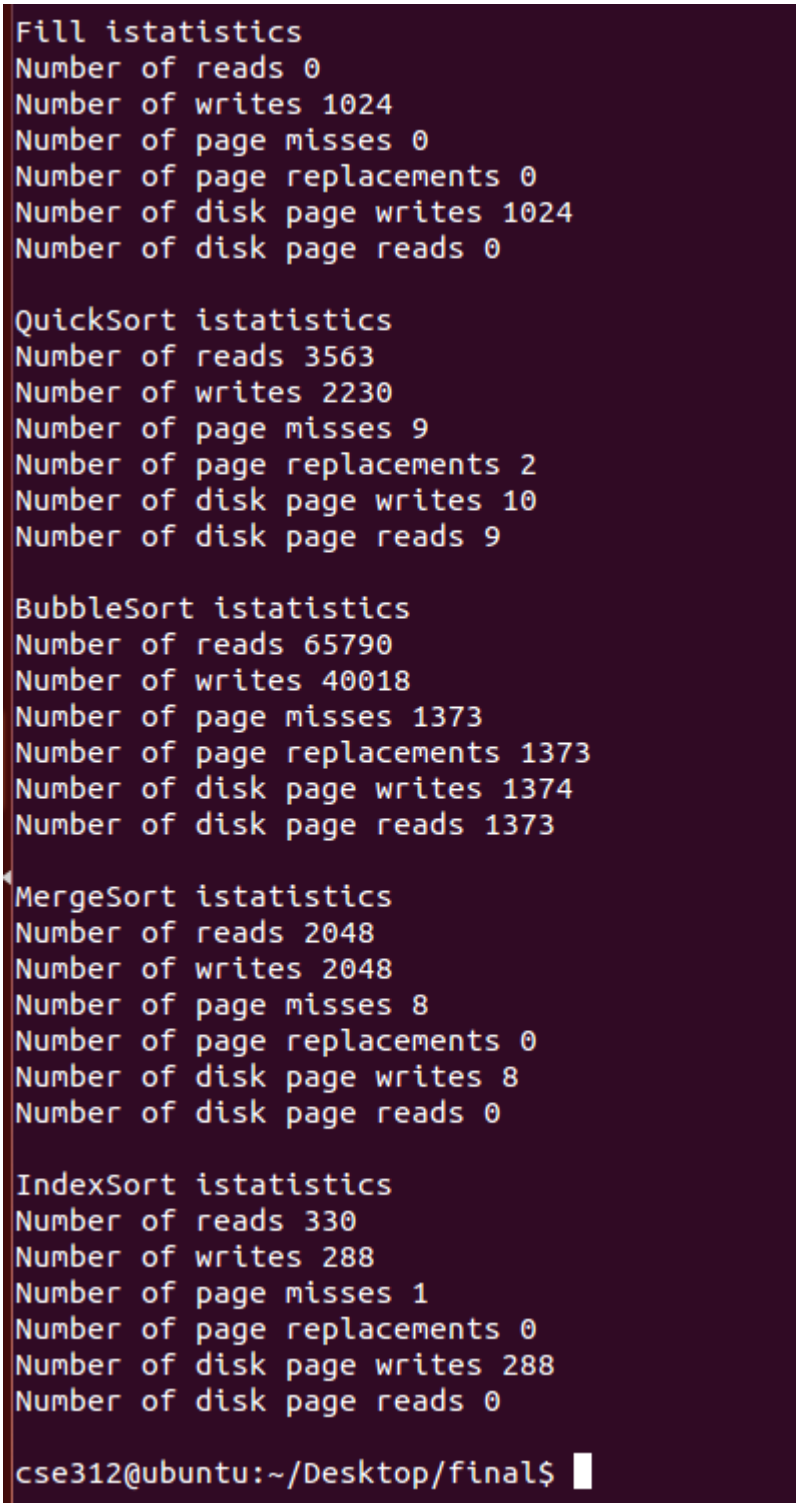
NRU is the last algorithm.According to modified and referenced field of pageframe decide index to replacement.I initialize another thread wthich name is refreshReferenced for reset the reference field 0.When finish the sorting algorithm, I set the finishControl value for finish thread process. Every 20 milliseconds referenced field will be zero with this another helper thread.If only page replacement type “NRU”, this thread created.If replacement algorithm does not NRU, this thread not creating.

```
void* refreshReferenced(void *a){
    unsigned int mSeconds = 20;
    int returnCode,i=0;

    while(finishControl==1){
        returnCode = usleep(mSeconds);
        for(i=0;i<numVirtual;i++){
            virtualAddressSpace[i].referenced=0;
        }
    }
    pthread_exit(NULL);
}
```

3 RESULT

3.1 Running Results



When finishing the program,end of the output like that.

```
Page table frame 27
Page table index 0
Page table referenced 0
Page table modified 0
Page table absent 0

Page table frame 28
Page table index 0
Page table referenced 0
Page table modified 0
Page table absent 0

Page table frame 29
Page table index 0
Page table referenced 0
Page table modified 0
Page table absent 0

Page table frame 30
Page table index 0
Page table referenced 0
Page table modified 0
Page table absent 0

Page table frame 31
Page table index 0
Page table referenced 0
Page table modified 0
Page table absent 0



Fill istatistics
Number of reads 0
Number of writes 1024
Number of page misses 0
Number of page replacements 0
Number of disk page writes 1024
Number of disk page reads 0

QuickSort istatistics
Number of reads 3563
Number of writes 2230
Number of page misses 9
Number of page replacements 2
Number of disk page writes 10
Number of disk page reads 9
```

Eample of pagetable field output.