МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Кафедра комп’ютерної інженерії та системного програмування

Звіт

про виконання лабораторної роботи №3

із дисципліни: «Системне програмне забезпечення»

на тему «Процеси в ОС Linux. Не витісняюче планування процесів»

Варіант №5

Виконала:

Студентка групи КІ2-19-1

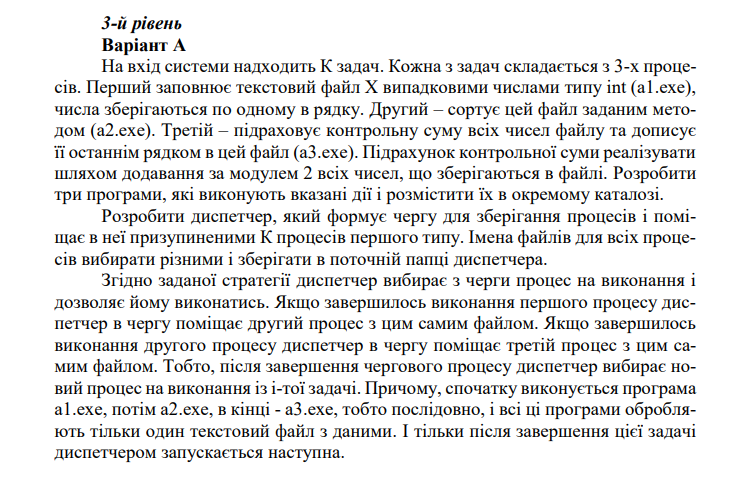
Гоба Каріна

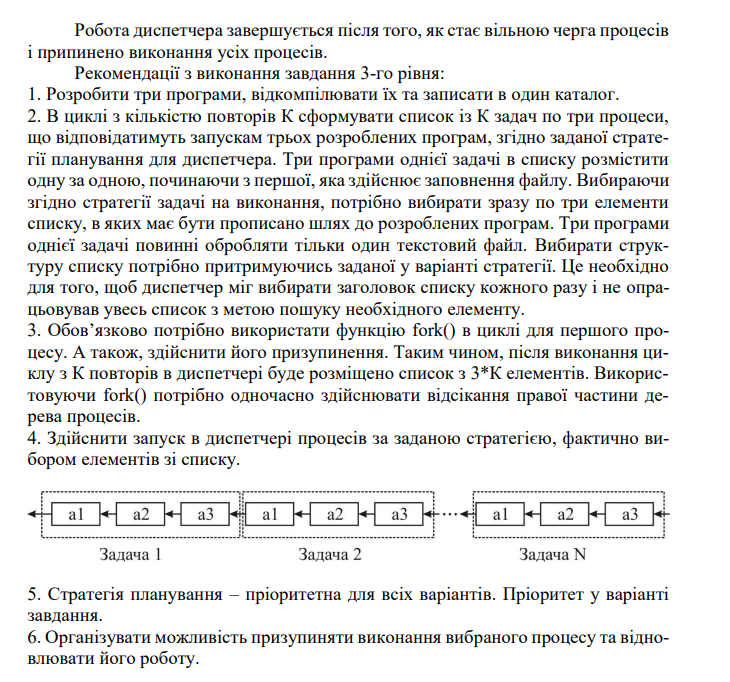
Перевірив:

Нічепорук А.О.

Хмельницький

2021 р.







Програма а1:

#include <QtCore/QCoreApplication>

#include<iostream>

#include<fstream>

#include<signal.h>

#include<time.h>

#include<stdlib.h>

#include<unistd.h>

#include<string>

using namespace std;

int main(int argc, char \*argv[])

{

int temp1; //count

char temp2[20];//name file

temp1=atoi(argv[1]);

sprintf(temp2,"%s",argv[2]);

srand(time(NULL));

ofstream f;

f.open(temp2,std::fstream::app);

f<<atoi(argv[1])<<endl;

for (int i=0;i<temp1;i++)

f<<rand()%(temp1+100)<<endl;

f.close();

exit(0);

}

Програма а2:

#include <QtCore/QCoreApplication>

#include<iostream>

#include<fstream>

#include<signal.h>

#include<time.h>

#include<stdlib.h>

#include<unistd.h>

#include<string>

using namespace std;

void poparpersort(int n, int ar[])

{

int tmp;  
   for(int i = 0; i <= n-1; i++)  
   {  
      for( int j = 0; j <= n-2-i; j++)  
      {  
          if( ar[j] > ar [j+1] )  
          {  
              tmp = ar[j];  
              ar[j] = ar[j+1];  
              ar[j+1] = tmp;  
          }  
      }  
  }

}

int main(int argc, char \*argv[])

{

//int temp1; //count

char temp2[20];//name file

//temp1=atoi(argv[1]);

sprintf(temp2,"%s",argv[1]);

ifstream f(temp2);

f>>N;

mas=new int[N];

for (int i=0;i<N;i++)

f>>mas[i];

poparpersort(0,N);

f.close();

ofstream p(temp2);

p<<N<<endl;

for (int i=0;i<N;i++)

p<<mas[i]<<endl;

p.close();

}

Програма а3:

#include <QtCore/QCoreApplication>

#include<iostream>

#include<fstream>

#include<signal.h>

#include<time.h>

#include<stdlib.h>

#include<unistd.h>

#include <string>

using namespace std;

int main(int argc, char \*argv[])

{

//int temp1; //count

char temp2[20];//name file

//temp1=atoi(argv[1]);

sprintf(temp2,"%s",argv[1]);

int N,\*mas,sum=0;

ifstream f(temp2);

f>>N;

mas=new int[N];

for (int i=0;i<N;i++)

{

f>>mas[i];

sum=sum+mas[i];

}

f.close();

ofstream p(temp2);

p<<N<<endl;

for (int i=0;i<N;i++)

p<<mas[i]<<endl;

p<<sum<<endl;

p.close();

}

Диспетчер:

//диспетчер FIFO

#include <stdio.h>

#include <signal.h>

#include <sys/time.h>

#include <math.h>

#include<time.h>

#include<unistd.h>

#include<sys/wait.h>

#include<stdlib.h>

#include<string.h>

#include<iostream>

using namespace std;

struct cherg

{

int pi;

int type;

int num;//number in order in the list

char fileName[20];

int countElem;

cherg \*next,\*pred;

};

int stat;

void addqueu(cherg \*\*head, int pid,int type,int num,char \*filename, int countElem)

{

cherg\* cur=(\*head);

cherg\* tmp=(cherg\*)malloc(sizeof(cherg));

if ((\*head) != NULL)

{

while (cur->next != NULL)//шукаємо останій елемент списку

cur = cur->next;

tmp->pi=pid;

tmp->type=type;

tmp->num=num;

tmp->countElem=countElem;

strcpy(tmp->fileName,filename);

cur->next = tmp;

tmp->pred=cur;

tmp->next = NULL;

//return head;

}

else

{

(\*head) = tmp;

(\*head)->next = NULL;

(\*head)->pred-NULL;

(\*head)->pi=pid;

(\*head)->type=type;

(\*head)->num=num;

(\*head)->countElem=countElem;

strcpy((\*head)->fileName,filename);

}

}

void addProcess(cherg \*\*head,int num,int type, char \*file,int countElem)

{

char filename[20]="",buf[20]="",buf1[10]="";

//int countElem=1;

if (type==1)

{

//формуємо назву файлу і кількість елементів для запису у файл: buf1,filename

countElem=rand()%100;

sprintf(buf1,"%d",countElem);//для конвертації з int в char\*

strcat(filename,"file\_#");// file\_#

sprintf(buf,"%d",num);//num in buf

strcat(filename,buf);// file\_#num

strcat(filename,".txt"); //file\_#num.txt

pid\_t pid = fork();

if (!pid) {//child

//cout<<getpid()<<endl;

kill(getpid(),SIGSTOP);//створення процесу сплячим

execl("a1","a1",buf1,filename,NULL);

exit(0);//вихід з дочірнього процесу

}

if (pid){//процес-батько

sleep(1);

addqueu(head,pid,type,num,filename,countElem);//батьку вертається pid, його і решту інфи заносим в //чергу

}

}

if (type==2)

{

pid\_t pid = fork();

if (!pid) {//child

//cout<<" "<<getpid()<<endl;

kill(getpid(),SIGSTOP);//create sleeping process

execl("a2","a2",file,NULL);//тут використано file (а не fileName), передається в параметри функції

exit(0);

}

if (pid){//parent

sleep(1);

addqueu(head,pid,type,num,file,countElem);

}

}

if (type==3)

{

pid\_t pid = fork();

if (!pid) {//child

// cout<<" "<<getpid()<<endl;

kill(getpid(),SIGSTOP);//create sleeping process

execl("a3","a3",file,NULL);

exit(0);

}

if (pid){//parent

sleep(1);

addqueu(head,pid,type,num,file,countElem);

}

}

}

int select\_proc(cherg \*\*head,int \*type, char \*fileName, int \*countElem, int \*num)//видалення з початку + //повернення pid видаленого елементу, кількості елементів, файлу, типу

{

int pi;

//розбір отриманого елементу на складові

pi=(\*head)->pi;

\*type=(\*head)->type;

strcpy(fileName,(\*head)->fileName);

\*countElem=(\*head)->countElem;

\*num=(\*head)->num;

if ((\*head)->next==NULL)//only one element

{

delete (\*head);//delete him

(\*head)=NULL;

}

else

{

(\*head)=(\*head)->next;//head element is next element

delete (\*head)->pred;//delete previous head

(\*head)->pred=NULL;

}

return pi;

}

int main() {

cherg \*head=NULL;

int type=1,num;//спочатку всі першого типу

char fileName[20];

int countElem;

int pi;

srand(time(NULL));

//штампуємо 3 процеси першого типу (максимальний розмір черги 3)

for(int i=0;i<3;i++)

{

addProcess(&head,i,type,fileName,countElem);

}

//поки не кінець черги

while(head != NULL){

pi=select\_proc(&head,&type,fileName,&countElem,&num);//отримали під, файл, тип, номер видаленого //елементу з черги

kill(pi,SIGCONT);//йому треба продовжити своє виконання

waitpid(pi,&stat,WCONTINUED | WSTOPPED);//чекаємо поки нащадок завершиться

cout<<"execute "<<num<<" "<<pi<<" "<<type<<endl;

sleep(1);

//відповідно 1 вже виконався, треба додати до черги тип 2, з тим самим файлом і кількістю елментів //(береться з select\_proc)

if (type!=3)

{

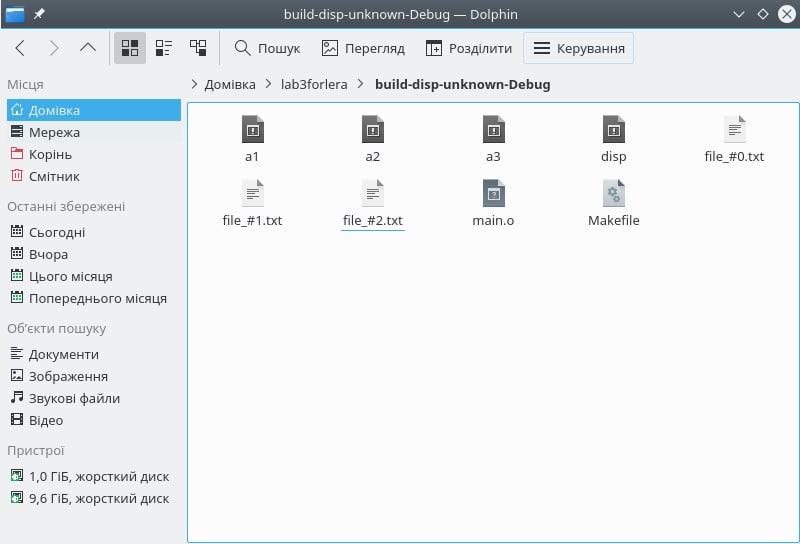
addProcess(&head,num+3,type+1,fileName,countElem);//додаємо тип 2 а потім тип 3

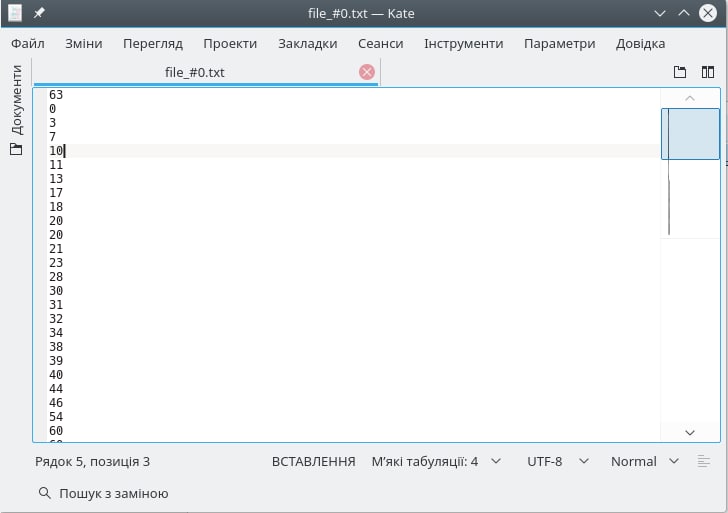
};

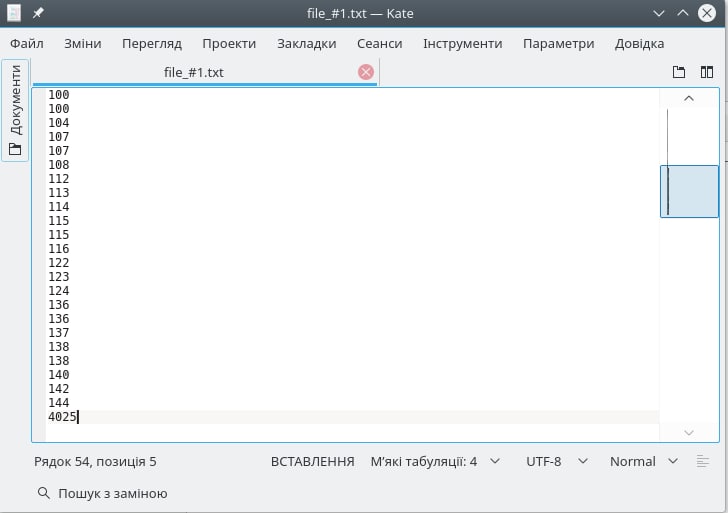
}

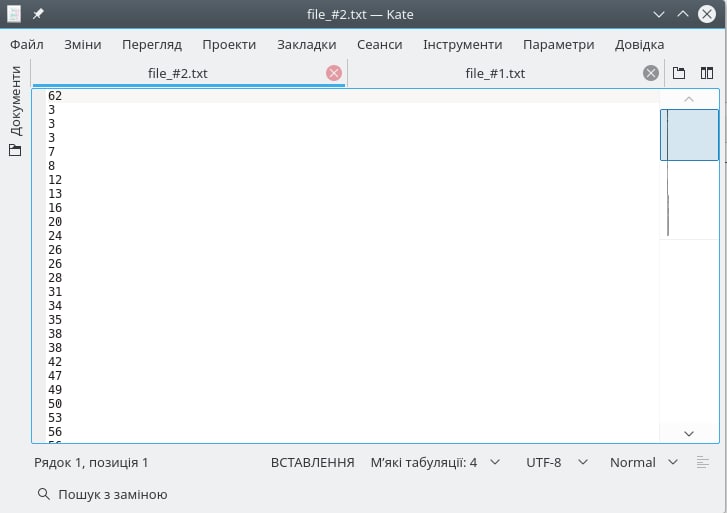
return 0;

}









Висновок: в даній лабораторній роботі було зроблено програму диспечер. Виконавши дану лабораторну роботу, я навчилась керувати роботою процесів у ОС Linux та ознайомилась із поняттям процесу.