Навчальна практика з операційних систем та системного програмування

Метою навчальної практики є формування у студентів здатності володіти знаннями щодо принципів роботи операційних систем, мати навички керування ресурсами обчислювальних систем, взаємодії з прикладним програмним забезпеченням, а також уміти обґрунтовано вибрати операційну систему для вирішення певних завдань і професійно її налаштовувати, уміти керувати розподіленими ресурсами обчислювальної системи.

В результаті проходження практики студент повинен отримати знання та навички:

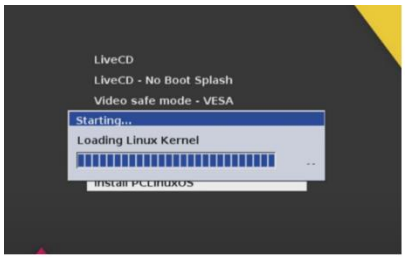
- з інсталяції і використання операційних систем GNU/Linux і Windows, в тому числі у віртуалізованих середовищах;
- з використання базового набору системних засобів операційної системи Linux для дослідження її поточного стану і керування обчислювальним процесом;
- з основ побудови операційних систем, їхньої архітектури, вимог до них, про історію їх розвитку і сучасні підходи до їх реалізації;
- про базовий склад компонентів операційної системи, основні функції ядра і системного програмного забезпечення;
- про методи і алгоритми керування локальними ресурсами комп'ютера: процесором, пам'яттю, пристроями введення-виведення, поділюваними ресурсами;
- про способи і засоби розв'язання проблем синхронізації і взаємних блокувань у багатозадачних і багатопотокових операційних системах;
- про принципи реалізації файлових систем, структуру сучасних файлових систем;
- про проблеми реалізації мережних функцій операційних систем і способи організації віддаленого виклику процедур і розподілених файлових систем;
- про підходи до реалізації зазначених вище механізмів у сучасних операційних системах GNU/Linux і MS Windows.

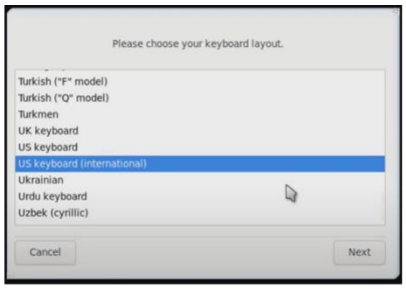
Під час практики студенти під керівництвом викладача знайомляться та/або опрацьовують різноманітні теми з операційних систем та системного програмування.

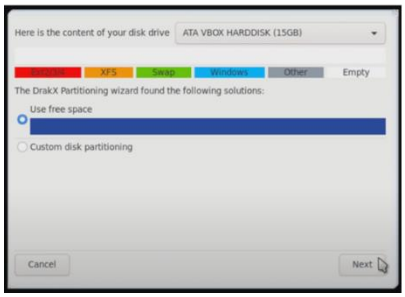
Орієнтовний зміст навчального дня практики:

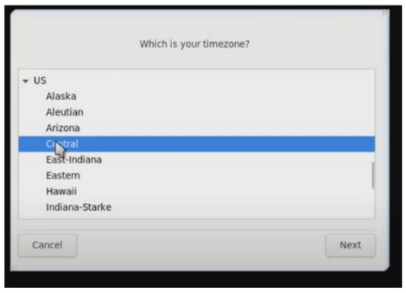
- обговорення з керівником практики поточної теми;
- індивідуальне опрацювання спеціалізованих задач згідно плану практики;
- документальне оформлення ходу розв'язання завдань.

Iz выполненных звітів робіт студентів:










```bash
#!/bin/bash
M=`expr $1 % 3`
if [ $M -eq 0 ]
then
    echo "ez a szam oszthato harommal"
    I=1
    F=1
    while test $1 -ge $I
    do
        F=`expr $F '*' $I`
        I=`expr $I '+' 1`
    done
    echo "a szam faktorialisa: $1! = $F"
else
    echo "ez a szam nem oszthato harommal"
fi
```

```bash
#!/bin/bash
M=`expr $1 % 3`
if [ $M -eq 0 ]
then
    echo "ez a szam oszthato harommal"
    I=1
    F=1
    while test $1 -ge $I
    do
        F=`expr $F '*' $I`
        I=`expr $I '+' 1`
    done
    echo "a szam faktorialisa: $1! = $F"
else
    echo "ez a szam nem oszthato harommal"
fi
```

```bash
#!/bin/bash
function zvit()
{
    for ((i=1;i<$1+1;i++))
    do
        echo -n "$i. szam: "
        read TOMB[$i]
    done
    for ((i=1;i<$1;i++))
    do
        for ((j=$i;j>1;j--))
        do
            if [ ${TOMB[$j+1]} -lt ${TOMB[$j]} ]
            then
                A=${TOMB[$j]}
                TOMB[$j]=${TOMB[$j+1]}
                TOMB[$j+1]=$A
            fi
        done
    done
    echo -n "Az elemek rendezve:"
    for ((i=0;i<$1;i++))
    do
        if [ ${TOMB[$i]} -lt 0 ]
        then
            echo -n "${TOMB[$i]} "
        fi
    done
    echo " "
}
zvit
```

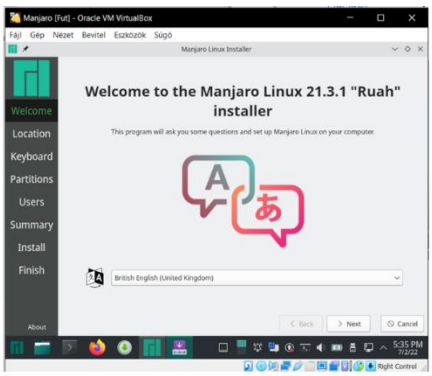
```bash
#!/bin/bash
DB=0
s=0
echo "Add meg az intervallum kezdo erteket:"
read A
echo "Add meg az intervallum vegerteket: "
read B
echo "Az intervallumon belul ez(ek) a szam(ok) oszthatok harommal es ottel is: "
for ((i=A;i<B;i++))
do
    M=`expr $i % 3`
    L=`expr $i % 5`
    if [ $M -eq 0 -a $L -eq 0 ]
    then
        echo -n "$i "
        DB=`expr $DB '+' 1`
        s=`expr $s '+' $i`
    fi
done
echo " "
echo "A szamok darabszama: $DB"
echo "A szamok osszege: $s"
```

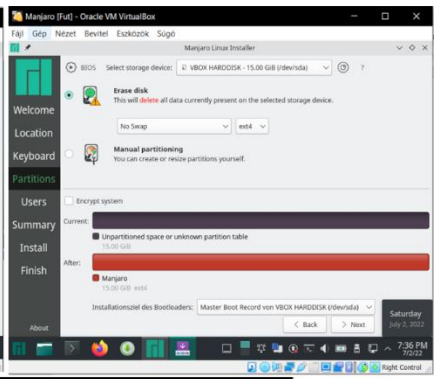
```bash
#!/bin/bash
for i in `seq 1 1 10`
do
    echo "$i"
    cd /home/beresn/Scriptek
    mkdir "$i"
done
cd /home/beresn/Scriptek
mkdir "`date`"
mkdir "BeresNoemi"
```

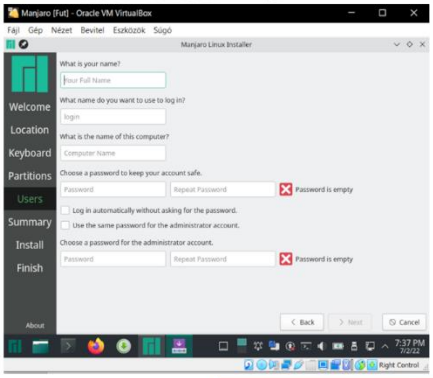
```bash
#!/bin/bash
if [[ `expr $1+$2-$3` -eq 0 ]]
then
    echo "Nullaval nem lehet osztani!"
else
    egy=`expr "scale=4; (($1^2)*$2*$3)/($1+$2-$3)*2*4*(a(1))" | bc -l`
    echo "az eredmeny: $egy"
fi

if [[ $1 -gt $2 ]]
then
    ketto=`expr $1 '%' $2`
    echo "az elso parameter nagyobb"
    echo "az elso es masodik szam maradekos osztasa: $ketto"
fi
```

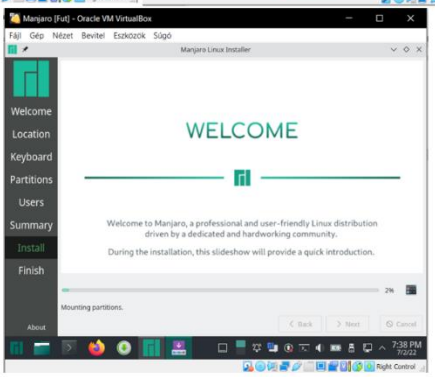
```bash
#!/bin/bash
echo "Adj meg egy szamot!"
read A
N=1
D=0
while test $N -le $(($A / 2))
do
    if test $(($A % $N)) -eq 0
    then
        D=$(($D + 1))
    fi
    N=$((N + 1))
done
if test $D -eq 1
then
    echo "A $A prímszám"
else
    echo "A $A nem prímszám"
fi
```

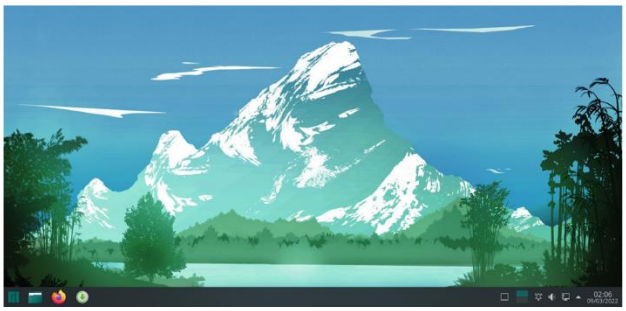
```bash
#!/bin/bash
for i in `seq 1 $1`
do
echo -n "$i. szam: "
read TOMB[$i]
done
echo " A tomb $1 elembol all: ${TOMB[*]}"
MIN=${TOMB[1]}
H=2
until test $H -gt $1
do
if test ${TOMB[H]} -lt $MIN
then
MIN=${TOMB[H]}
fi
H=`expr $H '+' 1`
done
echo "A tomb legkisebb eleme: $MIN"
```











```bash
#!/bin/bash
echo "adj meg egy szamot"
read n
while [ $n -lt 3 -o $n -gt 10000 ]
do
    echo "masik szamot adj meg"
    read n
done
if [ $(($n%3)) -eq 0 -a $(($n%5)) -eq 0 ]
then
    echo "A szam osztodik 3-mal es 5-tel"
else
    echo "A szam nem osztodik 3-mal es 5-tel"
fi
```

```bash
#!/bin/bash
declare -a array
n=$1
for ((i=0;i<$n;i++))
do
    read m
    array[$i]=$m
done
echo "${array[*]}"
max=${array[0]}
for ((i=1;i<$n;i++))
do
    if [ ${array[i]} -gt $max ]
    then
        max=${array[i]}
    fi
done
echo "maximum = $max"
```

```bash
#!/bin/bash
x=$1
if [ $x -eq 0 ]
then
    echo "nem lehet nulla"
else
    F=`echo "scale=10; ((c(3*$x)^2)/(2*$x))+((s(5*$x)/c(5*$x))^(1/5))" | bc -l`
    echo "a kifejezes erteke: $F"
fi
```

```bash
#!/bin/bash
n=$1
if [ $(($n%2)) -eq 0 ]
then
    echo "paros"
else
    echo "paratlan"
    m=$(($n%3))
    echo "3mal osztva a maradek: $m"
fi
```

```bash
#!/bin/bash
n=$1
f=1
if [ $n -ge 0 ]
then
    for ((i=1;i<=$n;i++))
    do
        f=$(($f*i))
    done
    echo "$n faktorialisa: $f"
else
    for ((i=-1;i>=$n;i--))
    do
        f=$(($f*i))
    done
    echo "$n faktorialisa: $f"
fi
```

```bash
#!/bin/bash

function tomb() {
    array=(1 4 6 2 3 7)
    S=1
    n=${#array[@]}
    for (( i=0; i<n; i++))
    do
        j=$(($i+1))
        if test $j -lt ${#array[@]}
        then
            K=$((${array[$i]}-${array[$j]}))
            S=$(($S*$K))
            #echo "$K"
            #echo "$S"
        else
            break
        fi
    done
    echo "$S"
}
tomb
```

```bash
#!/bin/bash
a=$1
b=$2
c=$3
s=`echo "scale=10; ((($a^2)*$b*$c)/($a+$b-$c))*s(4*a(1))" | bc -l`
if [ $a -gt $b ]
then
    d=$(($a%$b))
    echo "maradek: $d"
fi
echo "eredmeny: $s"
```


```
#!/bin/bash
function maxesminatlag() {
echo -n "adja meg a tömb hosszát: "
read n
i=0
while [ $i -lt $n ]
   do
   tomb[$i]=$(( $RANDOM % 100 + 1 ))
   i=`expr $i + 1`
   done
echo "a tömb elemei: ${tomb[*]}"
min=${tomb[0]}
max=${tomb[0]}
j=1
until [[ $j -ge $n ]]
   do
   if [[ ${tomb[j]} -lt $min ]]
      then
      min=${tomb[j]}
   fi
   if [[ ${tomb[j]} -gt $max ]]
      then
      max=${tomb[j]}
   fi
   j=`expr $j '+' 1`
   done
echo "a legkisebb elem: $min"
echo "a legnagyobb elem: $max"
echo "átlag: `echo "scale=1; ($min+$max)/2" |bc`"
}
maxesminatlag
```

```
#!/bin/bash
if test $# -ne 3
  then
  echo "Három paramétert adj meg!"
  exit 0
fi
tomb=($1 $2 $3)
max=${tomb[0]}
i=1
until test $i -gt 2
   do
   if test ${tomb[i]} -gt $max
      then
      max=${tomb[i]}
   fi
   i=`expr $i '+' 1`
   done
echo "A paraméterek között a legnagyobb: $max"

if test $max -lt 0
  then
  echo "Negatív számból nincs faktoriális"
  exit 0
fi
j=1
f=1
while test $max -ge $j
  do
  f=`expr $f '*' $j`
  j=`expr $j + 1`
  done
echo "$max faktoriálisa: $f"
```

```
#!/bin/bash
if [[ $# -ne 3 ]]
  then
  echo "Három paramétert adj meg!"
  exit 0
fi
tomb=($1 $2 $3)
min=${tomb[0]}
for (( i=0; i<3; i++ ))
do
  if [[ ${tomb[i]} -lt $min ]]
  then
  min=${tomb[i]}
  fi
done
echo "A paraméterek között a legkisebb: $min"
if [ $min -lt 0 ]
  then
  echo "Negatív számból nincs faktoriális"
  exit 0
fi
F=1
for (( i=1; i<=$min; i++ ))
do
  F=`expr $F '*' $i`
done
echo "$min faktoriálisa: $F"
```