Operációs rendszerek BSc

9. Gyak. 2022. 04. 04.

Készítette:

Juhász Balázs Bsc Mérnökinformatikus ZUYISF

Miskolc, 2022

1. feladat –

A tanult rendszerhívásokkal (open(), read()/write(), close()-ők fogják a rendszerhívásokat tovább hívni - írjanak egy neptunkod_openclose.c programot, amely megnyit egy fájlt - neptunkod.txt, tartalma: hallgató neve, szak, neptunkod.

A program következő műveleteket végezze:

- olvassa be a neptunkod.txt fájlt, melynek attribútuma: O_RDWR
- hiba ellenőrzést,
- write() mennyit ír ki a konzolra.

- read() kiolvassa a neptunkod.txt tartalmát és mennyit olvasott ki (byte), és kiírja konzolra.
- lseek() pozícionálja a fájl kurzor helyét, ez legyen a fájl eleje: SEEK_SET, és kiírja a konzolra.

```
int main()

int fileHandle = open(FILE, 0_RDWR);
if(fileHandle == -1)

perror("Nem sikerult megnyitni a fajlt!");
return 1;

else
{
    printf("Sikeres volt a fajl megnyitasa!\n");
}
char tartalom[120];
int olvasott = read(fileHandle, tartalom, sizeof(tartalom));
printf("Reolvasott tartalom: \"%s\" osszesen: \"%i\" byte.\n",tartalom,olvasott);

lseek(fileHandle, 0, SEEK_SET);

char text[] = "teszt";
int irt = write(fileHandle, text, sizeof(text));
printf("A fajlba irtuk a(z) \"%s\" szoveget. Osszesen: \"%i\" byte. \n", text, irt);
close(fileHandle);

return 0;
}
```

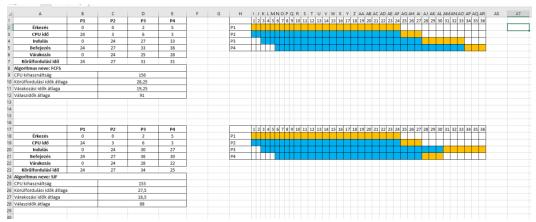
2. feladat –

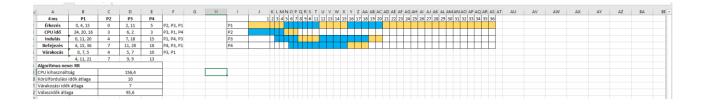
Készítse el a következő feladatot, melyben egy szignálkezelő több szignált is tud kezelni:

- **a.**) Készítsen egy szignál kezelőt (handleSignals), amely a SIGINT (CTRL + C) vagy SIGQUIT (CTRL + \) jelek fogására vagy kezelésére képes.
- **b.**) Ha a felhasználó SIGQUIT jelet generál (akár kill paranccsal, akár billentyűzetről a CTRL + \) a kezelő egyszerűen kiírja az üzenetet visszatérési értékét a konzolra.
- **c.**) Ha a felhasználó először generálja a SIGINT jelet (akár kill paranccsal, akár billentyűzetről a CTRL + C), akkor a jelet úgy módosítja, hogy a következő alkalommal alapértelmezett műveletet hajtson végre (a SIG_DFL) kiírás a konzolra.

```
void handleSignals(int signum);
     if(sigHandlerReturn == SIG_ERR)
          perror(*Signal error*);
return 1;
     sigHandlerReturn = signal(SIGQUIT, sigHandlerQuit);
     if(sigHandlerInterrupt == SIG_ERR)
           perror("Signal error");
return 1;
           printf("A program leallitasahoz a kovetkezoket vegezze el: \n");
printf("1. Nxisson meg eny másik terminalt.\n");
printf("2. Adja ki a parancsot: kill: %d \n". getpid());
sleep(10);
     void handleSignals(int signum)
             switch(signum)
                    case SIGINT:
    printf("\n CTRL+C-t eszlelt\n");
    signal(SIGINT, SIG_DFL);
    break;
case SIGQUIT:
    printf("SIQUIT aktivalodott\n");
    break;
default:
                             printf("\nEogadott jel szama: %d\n", signum);
break;
             return ;
                                 orogram leallitasahoz a kovetkezoket vegezze el:
Nyisson meg egy másik terminalt.
Adja ki a parancsot: kill: 1128
```

3. feladat – Adott a következő ütemezési feladat, amit a FCFS, SJF és Round Robin (RR: 4 ms) ütemezési algoritmus alapján határozza meg következő **teljesítmény értékeket, metrikákat** (külön-külön táblázatba)





Gyakorló feladat:

2. feladat – Írjon C nyelvű programot, amelyik kill() seg.-vel SIGALRMet küld egy argumentumként megadott PID-u processznek, egy másik futó program a SIGALRM-hez rendeljen egy fv.-t amely kiírja pl. neptunkodot, továbbá pause() fv.-el blokkolódjon, majd kibillenés után jelezze, hogy kibillent és terminálódjon.

```
#include <stdio.h>
  #include <unistd.h>
  #include <stdlib.h>
  #include <sys/types.h>
  #include <string.h>
  #include <signal.h>
  void handleSigalarm();
  int main()
= (
      printf("A program pidie: %d\n",getpid());
      signal(SIGALRM, handleSigalarm);
      pause();
      printf("Kibillent\n");
      exit(0);
      return 0;
  void handleSigalarm()
-(
```

3. feladat – Írjon C nyelvű programot, amelyik a SIGTERM-hez hozzárendel egy fv-t., amelyik kiírja az int paraméter értéket, majd végtelen ciklusban fusson, 3 sec-ig állandóan blokkolódva elindítás után egy másik shell-ben kill paranccsal (SIGTERM) próbálja terminálni, majd SIGKILL-el."

```
DLWGQZ_gyak9_2.c III
            #include <stdio.h>
     1
     2
            #include <sys/types.h>
     3
            #include <signal.h>
            #include <unistd.h>
     6 7
            void kezelo(int i)
     8
                 printf("Signal kezelese: %d\n",i);
     9
                 return;
    10
11
12
13
14
            int main()
          □{
                 printf("PID: %d\n",getpid());
printf("Signal kezelo atvetele: %d\n",signal(SIGTERM,&kezelo));
while(1)
    15
    16
    17
18
19
                       printf("lepes\n");
    20
21
                       sleep(3);
          }
    22
```