Operációs Rendszerek Bsc

3.gyak.

2021.03.03.

Készítette:

Horváth Ákos Bsc

Programtervező informatikus szak

R3SZY2

1.feladat-Adott négy processz a rendszerbe, melynek beérkezési sorrendje: A, B, C és D. Minden processz USER módban fut és mindegyik processz futásra kész.

Kezdetben mindegyik processz *p_uspri = 60*.

Az A, B, C processz *p_nice = 0*, a D processz *p_nice = 5*.

Mindegyik processz *p_cpu = 0*, az óraütés 1 indul, a befejezés legyen 201. óraütés-ig.

Határozza meg az ütemezést RR nélkül és az ütemezést RR-nal - külön-külön táblázatba.

RR nélkül:

Minden 100. Óraütésnél processzek p_cpu értékét beszoroztam a korrekciós faktorral.

$KF = 2 \cdot processzek\ száma/2 \cdot processzek\ száma+1$

Minden 100. Óraütésnél átszámoltam a processzek prioritását az alábbi képlet alapján.

*p_pri = P_USER + p_cpu / 4 + 2 * p_nice*

Lásd Github repositoryban *unix_utem_rrnelkul.xlsx* fájl.

Round robinnal:

Minden 100. Óraütésnél processzek p_cpu értékét beszoroztam a korrekciós faktorral.

$KF = 2 \cdot processzek\ száma/2 \cdot processzek\ száma+1$

Minden 100. Óraütésnél átszámoltam a processzek prioritását az alábbi képlet alapján.

*p_pri = P_USER + p_cpu / 4 + 2 * p_nice*

Ha azonos priorítású processzek voltak, 10 ms-ként váltakozott a futó processz.

Lásd Github repositoryban *unix_utem_rr.xlsx* fájl.

2.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.

Lásd Github repositoryban *r3szy2_openclose.c* fájl.

```
int file = open("r3szy2.txt", O_RDWR);
   if(file < 0)
   {
      printf("File megnyitasa sikertelen\n");
      return 1;
   }
```

Int file: file descriptor, ha open() visszatérési értéke kisebb mint 1, megnyitás sikertelen.

Argumentmai a fájl elérési útvonala, és megnyitási mód.

O_RDWR: olvasni és írni is tudunk a fájlba

```
bytesread = read(file, str, BYTE_COUNT);
   if(bytesread < 0)
   {
      printf("File olvasasa sikertelen\n");
      return 1;
   }

   printf("%d byte kiolvasva a fajlbol\n", bytesread);
```

int bytesread: hány bájtot olvastunk ki a fájlból, read() visszatérési értéke. Hasonlóan az open() függvényhez, negatív visszatérési érték hibát jelez.

Argumentumai: file descriptor, karaktertömb amibe beolvassa a fájlt, may hány bájtot olvasson be.

```
cursorpos = lseek(file, 0, SEEK_SET);

   if(cursorpos < 0)
   {
      printf("Kurzor allitasa sikertelen\n");
      return 1;
```

```
}

printf("Kurzor tavolsaga a fajl elejetol: %d\n", cursorpos);
```

Int cursorpos: kurzor távolsága a fájl elejétől számítva. Hasonlóan az open() és read() függvényhez, negatív visszatérési érték hibát jelez.

Argumentumai: file descriptor, kurzor pozíciója, kurzormozgatás módja.