

Operációs rendszerek Bsc

5. gyakorlat

2021. 03. 10.

Készítette:

Molnár Balázs Bsc

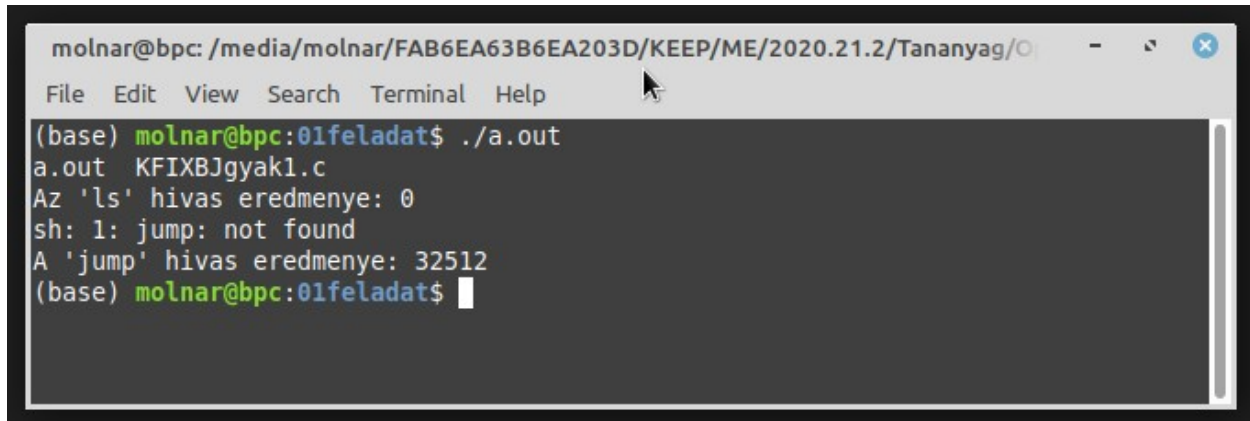
programtervező informatikus

KFIXBJ

Miskolc, 2021

1. feladat - A `system()` rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési értéket! Mentés: `neptunkodgyak1.c`

A 'KFIXBJgyak1.c'-ből fordított 'a.out' futtatásakor a következő eredményt kaptam:

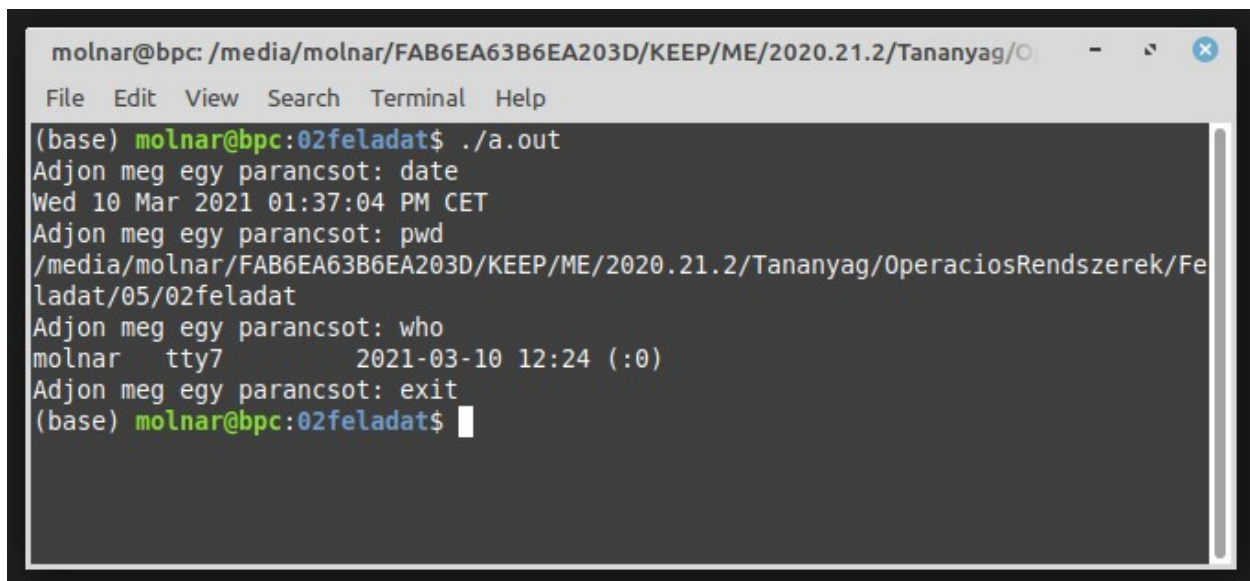


```
molnar@bpc: /media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/O
File Edit View Search Terminal Help
(base) molnar@bpc:01feladat$ ./a.out
a.out KFIXBJgyak1.c
Az 'ls' hívás eredménye: 0
sh: 1: jump: not found
A 'jump' hívás eredménye: 32512
(base) molnar@bpc:01feladat$
```

A létező parancs az 'ls' volt, ennek eredménye az első sorban látható ('a.out KFIXBJgyak1.c', a mappa fájljai), a visszatérési kód (0) a helyes futást jelzi.

A nem létező parancs a 'jump' volt, ennek eredménye: 'sh: 1: jump: not found', a visszatérési érték pedig az ehhez tartozó hibakód (32512).

2. feladat - Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre.



```
molnar@bpc: /media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/O
File Edit View Search Terminal Help
(base) molnar@bpc:02feladat$ ./a.out
Adjon meg egy parancsot: date
Wed 10 Mar 2021 01:37:04 PM CET
Adjon meg egy parancsot: pwd
/media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/OperaciosRendszerek/Feladat/05/02feladat
Adjon meg egy parancsot: who
molnar  tty7          2021-03-10 12:24 (:0)
Adjon meg egy parancsot: exit
(base) molnar@bpc:02feladat$
```

3. feladat - Készítsen egy parent.c és child.c programot. A parent.c elindít egy gyermek processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (5-ször)!



```
molnar@bpc: /media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/O - [x]
File Edit View Search Terminal Help
(base) molnar@bpc:03feladat$ ./parent
Molnar Balazs - KFIXBJ
Molnar Balazs - KFIXBJ
Molnar Balazs - KFIXBJ
Molnar Balazs - KFIXBJ
Molnar Balazs - KFIXBJ
child terminated
(base) molnar@bpc:03feladat$
```

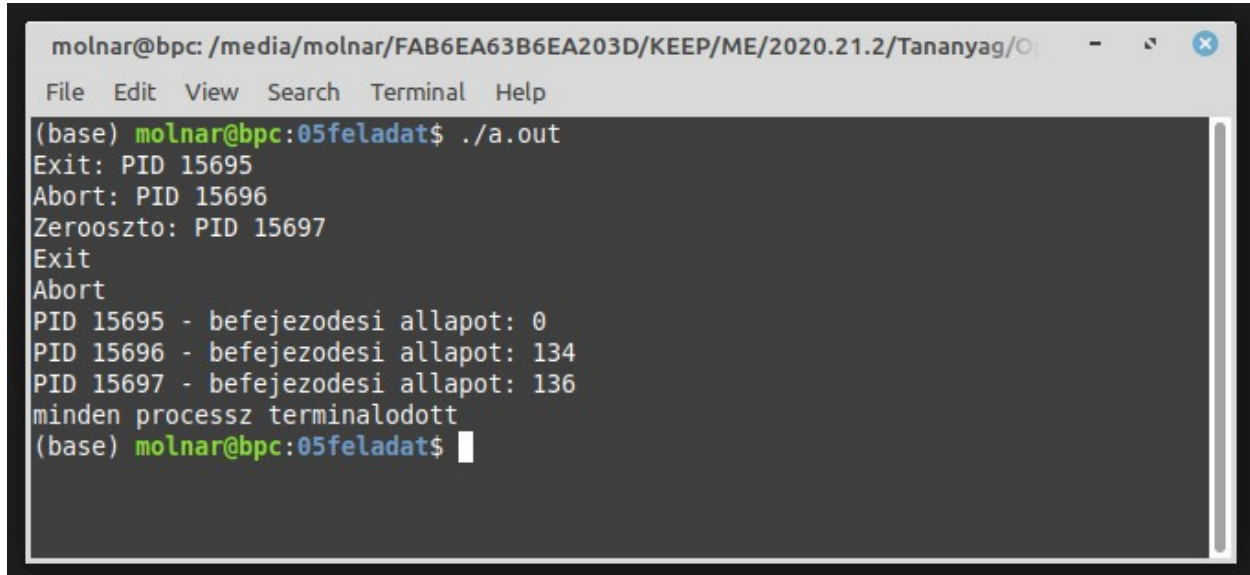
4. feladat - A fork() rendszerhívással hozzon létre egy gyerek processzt és abban hívjon meg egy exec családbeli rendszerhívást (pl. execlp). A szülő várja meg a gyerek futását!



```
molnar@bpc: /media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/O - [x]
File Edit View Search Terminal Help
(base) molnar@bpc:04feladat$ ./a.out
a.out KFIXBJgyak4.c
child terminated
(base) molnar@bpc:04feladat$
```

5. feladat - A `fork()` rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekekben: `exit`, `abort`, nullával való osztás)!

A következő eredményt kapjuk:



```
molnar@bpc: /media/molnar/FAB6EA63B6EA203D/KEEP/ME/2020.21.2/Tananyag/O
File Edit View Search Terminal Help
(base) molnar@bpc:05feladat$ ./a.out
Exit: PID 15695
Abort: PID 15696
Zeroosztó: PID 15697
Exit
Abort
PID 15695 - befejezodesi allapot: 0
PID 15696 - befejezodesi allapot: 134
PID 15697 - befejezodesi allapot: 136
minden processz terminalodott
(base) molnar@bpc:05feladat$
```

Az `Exit` és az `Abort` gyermekprocessz végrehajtódik, a zéróosztásnál nem kapunk kimenetet, mivel még a kiírás előtt hiba történik, a művelet elvégzésekor.

Ha kikeressük a gyakori hibakódokat, láthatjuk, hogy a gyermekprocesszek által visszaadott hibakódok pontosan megegyeznek.

1.4 Common Error Codes

This document outlines the meaning of several commonly-encountered error codes in Seashell.

- 1.4.1 Code 0 - Successful Completion
- 1.4.2 Error code 1 - General Error
- 1.4.3 Error code 134 - Program Abort
- 1.4.4 Error code 136 - Erroneous Arithmetic Operation
- 1.4.5 Error code 139 - Segmentation Fault
- 1.4.6 Error code 255 - Program Timed Out

6. feladat - Adott a következő ütemezési feladat, ahol a RR ütemezési algoritmus használatával készítse el:

- Ütemezze az adott időszelot alapján az egyes processzek paramétereit (ms)!
- A rendszerben lévő processzek végrehajtásának sorrendjét?
- Ábrázolja Gantt diagram segítségével az aktív/várákozó processzek futásának menetét!

5. gyakorlat

⑥ RR: 5 ms

a)	P1	P2	P3	P4	P5
Érkezés	0	1	3	9	12
CPU idő	3	8, 3	2	20, 15	5
Indulás	0	3, 10	8	13, 23	18
Befejezés	3	8, 13	10	18, 38	23
Várákozás	0	² (3), 2	⁵ (3)	⁴ (18), 5	⁶ (18)

b) Végrehajtási sorrend:

P1, P2, P3, P2, P4, P5, P4

