**Operációs rendszerek BSC**

2.Konzultáció

2022.03.18.

Készítette:

Horkai Adrián Attila

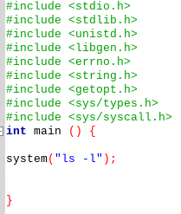
Neptun kód: MDC16I

# **1.** **feladat**

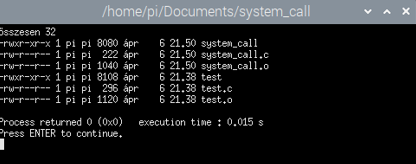
*A system() rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja  
a visszatérési érteket, magyarázza egy-egy mondattal*

Megvalósítás:

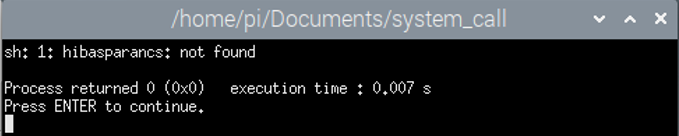
Az alábbi kódrészt használtam a system() parancsok megadására.



Érvényes parancs eredménye a következő:



Érvénytelen parancs eredménye pedig az alábbi képen látható:

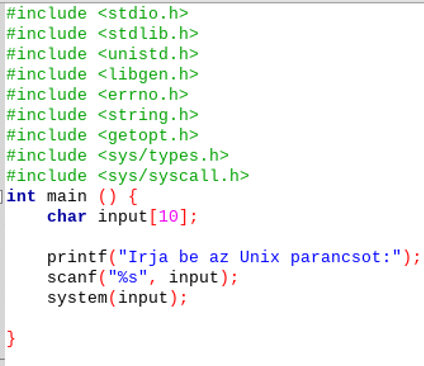


# **2.** **feladat**

*Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre. (pl.: amit bekér: date, pwd, who etc.; kilépés: CTRL-\) - magyarázza egy-egy mondattal.*

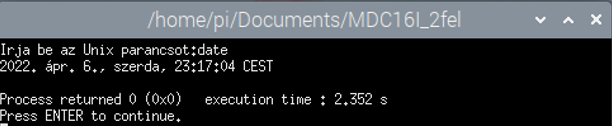
Megvalósítás:

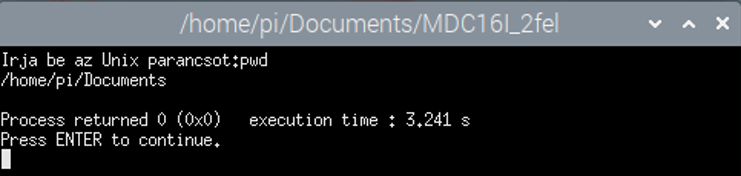
Az alábbi kódrészt használtam az Unix parancsok bekérésére:

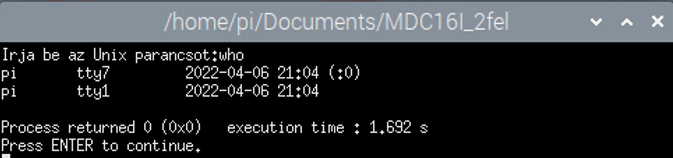


A linux konzolból - az operációs rendszeremen - a CTRL+C billentyűkombináció lenyomásával bezárható a felület.

A parancsok eredményei az alábbi képeken láthatóak:







A date parancs megadja az aktuális dátumot, a pwd parancs megadja az aktuális munkaknyvtárat, a who parancs pedig az aktuálisan bejelentkezett felhasználókat listázza.

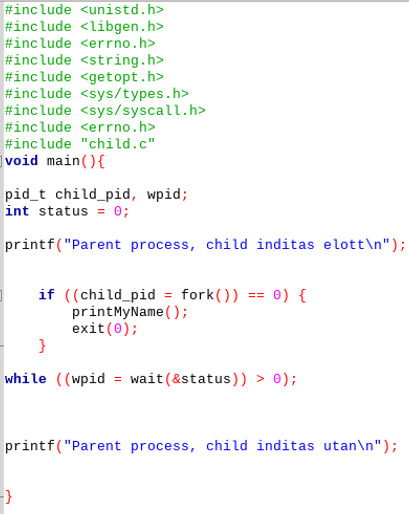
# **3.** **feladat**

*Készítsen egy parent.c és a child.c programokat. A parent.c elindít egy gyermek  
processzt, ami különbözik a szülőtől. A szülő megvárja a gyermek lefutását. A gyermek szöveget ír a szabványos kimenetre (10-ször) (pl. a hallgató neve és a neptunkód)!*

Megvalósítás:

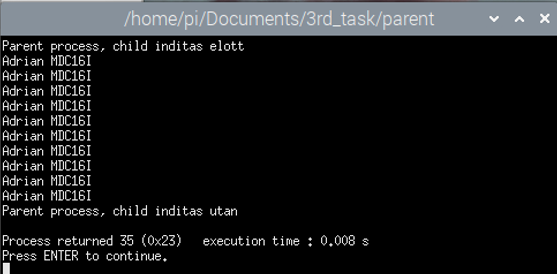
A megvalósításhoz az alábbi kódrészt használtam:

* parent.c program



A parent.c program a child.c-ben található printMyName() függvényt használja a név és neptun kód 10-szer kiírására (for ciklus használatával).

A wait parancs visszaadja a gyermek process aktuális státuszát, így egy while ciklus segítségével a child process lefutása után folytatja csak a parent process-t. Ennek figyelésére két printf-t alkalmaztam. A futtatás eredménye a következő képen látható:

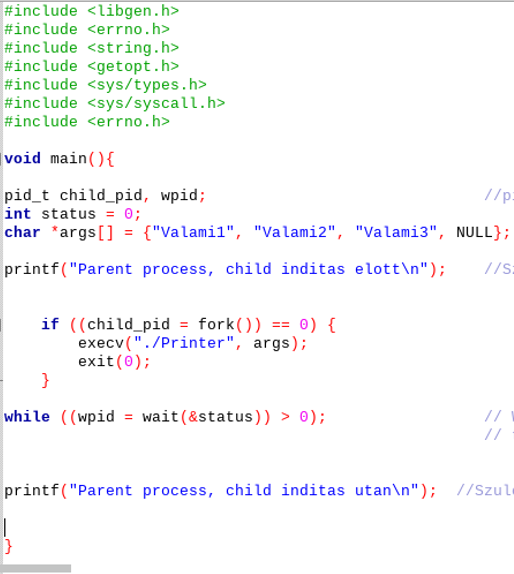


# **4.** **feladat**

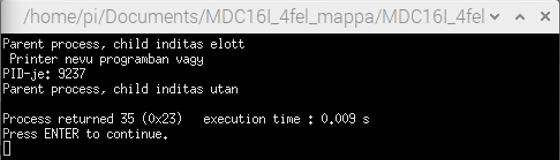
*A fork() rendszerhívással hozzon létre egy gyerek processzt-t és abban hívjon meg egy exec  
családbeli rendszerhívást (pl. execlp). A szülő várja meg a gyerek futását! - magyarázza egy-egy mondattal.*

Megvalósítás:

Az előző feladatban is alkalmazott fork() rendszerhívással létrehoztam egy gyermek process-t, majd az execv() paranccsal meghívtam a Printer program futtatható állományát, amely kiírja, hogy sikerült a process cseréje a Printer programban megadott printf-es kiírásra.



A futtatott program a következő képen látható:

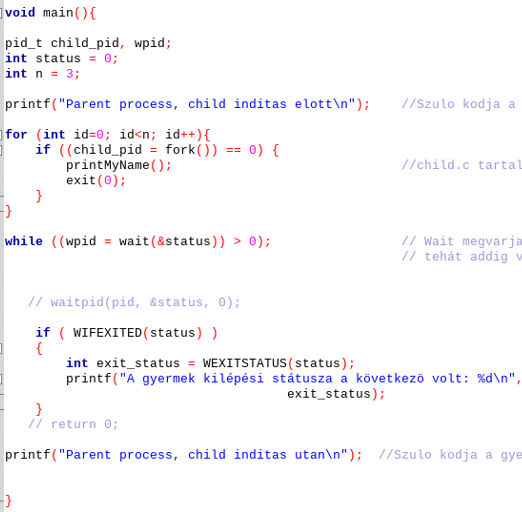


# **5.** **feladat**

*A fork() rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési  
állapotokat (gyerekben: exit, abort, nullával való osztás)! - magyarázza egy-egy  
mondattal!*

Megvalósítás:

A következő kódrészt használtam a megvalósításhoz.



Egy for ciklussal 3 gyermek processt hozok létre a fork() rendszerhívással. A while ciklusban a wait-tel figyelem a processek lefutásának állapotát.

A wifexited opciós argument true-t ad vissza, ha a gyermek terminálása megtörtént. Ha pedig ennek visszatérési értéke true, akkor a WEXITSTATUS-szal visszakérem a befejeződési állapot számkódját. Így figyelhetőek a befejeződési állapotok.

A programfutás eredménye a következő képen látható:

