

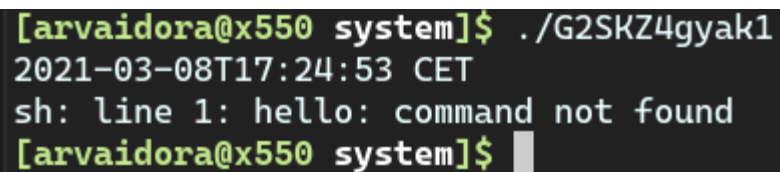
1. A `system()` rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési értéket!

G2SKZ4gyak1.c:

```
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
```

```
int main(void) {
    system("date");
    // nem létező
    system("hello");

    return 0;
}
```



```
[arvaidora@x550 system]$ ./G2SKZ4gyak1
2021-03-08T17:24:53 CET
sh: line 1: hello: command not found
[arvaidora@x550 system]$
```

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

G2SKZ4gyak2.c:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>

#define BUFFER_SIZE 255

int main(void) {
    char buffer[BUFFER_SIZE];
    do
    {
        printf(">");
        fgets(buffer, BUFFER_SIZE, stdin);
        system(buffer);
    }
    while (1);

    return 0;
}
```

```
[arvaidora@x550 system]$ ./G2SKZ4gyak2
>date
2021-03-08T17:26:36 CET
>ls
G2SKZ4gyak1      G2SKZ4gyak2      G2SKZ4gyak4      G2SKZ4gyak5      Makefile  child.c
G2SKZ4gyak1.c   G2SKZ4gyak2.c    G2SKZ4gyak4.c    G2SKZ4gyak5.c    child     parent.c
>tree
.
├── G2SKZ4gyak1
├── G2SKZ4gyak1.c
├── G2SKZ4gyak2
├── G2SKZ4gyak2.c
├── G2SKZ4gyak4
├── G2SKZ4gyak4.c
├── G2SKZ4gyak5
├── G2SKZ4gyak5.c
├── Makefile
├── child
├── child.c
└── parent.c

0 directories, 12 files
>^\Quit (core dumped)
[arvaidora@x550 system]$
```

3. 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 (5-ször) (pl. a hallgató neve és a neptunkód)!

child.c:

```
#include <stdio.h>
#include <unistd.h>

int main(void) {
    for (int i = 0; i < 5; i++)
    {
        printf("Árvai Dóra G2SKZ4\n");
        sleep(1);
    }

    return 0;
}
```

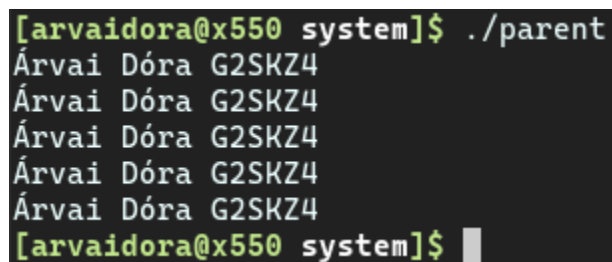
parent.c:

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>

int main(void) {
    int pid;

    if ((pid = fork()) < 0)
        perror("fork error");
    else if (pid == 0)
    {
        if (execl("./child", "child", (char *)NULL) < 0)
            perror("execl error");
    }
    if (waitpid(pid, NULL, 0) < 0)
        perror("wait error");

    return 0;
}
```



```
[arvaidora@x550 system]$ ./parent
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
[arvaidora@x550 system]$
```

4. 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!

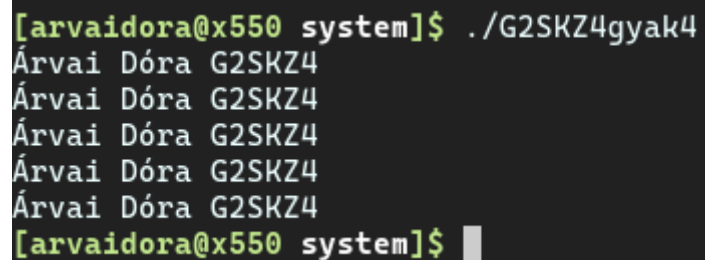
G2SKZ4gyak4.c:

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int pid;
    int status;

    if ((pid = fork()) < 0)
        perror("fork hiba");
    else if (pid == 0) /* gyermek */
        if (execl("./child", "child", (char *)NULL) < 0)
            perror("execl error");
    if (wait(&status) != pid)
        perror("wait hiba"); /*szülő */

    return 0;
}
```



```
[arvaidora@x550 system]$ ./G2SKZ4gyak4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
Árvai Dóra G2SKZ4
[arvaidora@x550 system]$
```

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

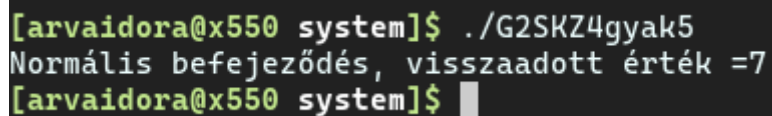
G2SKZ4gyak5.c:

```
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int pid;
    int status;

    if ((pid = fork()) < 0)
        perror("fork hiba");
    else if (pid == 0) /* gyermek */
        exit(7);      /* befejeződik */
    if (wait(&status) != pid)
        perror("wait hiba"); /*szülő */
    if (WIFEXITED(status))
        printf("Normális befejeződés, visszaadott érték =%d\n",
WEXITSTATUS(status));

    return 0;
}
```



```
[arvaidora@x550 system]$ ./G2SKZ4gyak5
Normális befejeződés, visszaadott érték =7
[arvaidora@x550 system]$
```