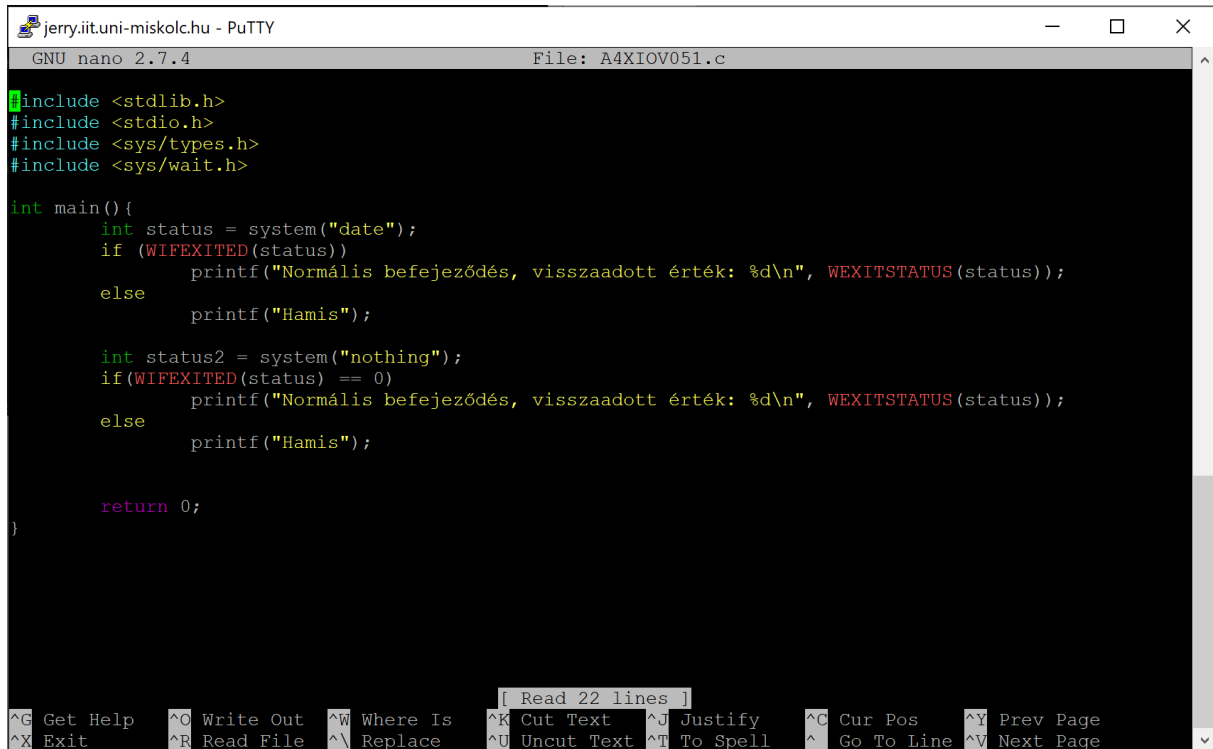


1. A `system()` rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési értéket! Mentés: *neptunkodgyak1.c*



```
GNU nano 2.7.4 File: A4XIOV051.c

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

int main() {
    int status = system("date");
    if (WIFEXITED(status))
        printf("Normális befejeződés, visszaadott érték: %d\n", WEXITSTATUS(status));
    else
        printf("Hamis");

    int status2 = system("nothing");
    if (WIFEXITED(status2) == 0)
        printf("Normális befejeződés, visszaadott érték: %d\n", WEXITSTATUS(status2));
    else
        printf("Hamis");

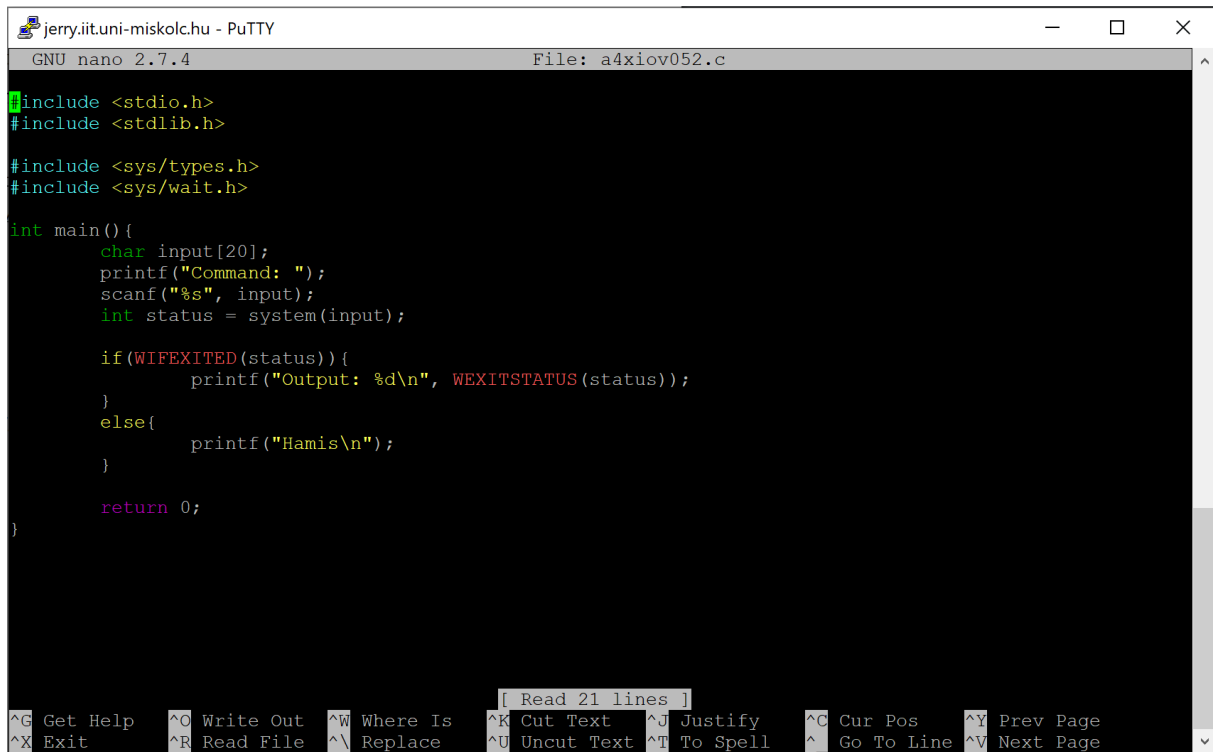
    return 0;
}
```

[Read 22 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page

```
abraham1@jerry:~$ ./a4xiov051.out
Tue Mar 16 18:59:48 CET 2021
Normális befejeződés, visszaadott érték: 0
sh: 1: nothing: not found
Hamis
```

2. Írjon programot, amely billentyűzetről bekér Unix parancsokat és végrehajtja őket, majd kiírja a szabványos kimenetre. (pl.: amit bekér: date, pwd, who etc.; kilépés: CTRL-\\) Mentés: *neptunkodgyak2.c*



```
GNU nano 2.7.4 File: a4xiov052.c

#include <stdio.h>
#include <stdlib.h>

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

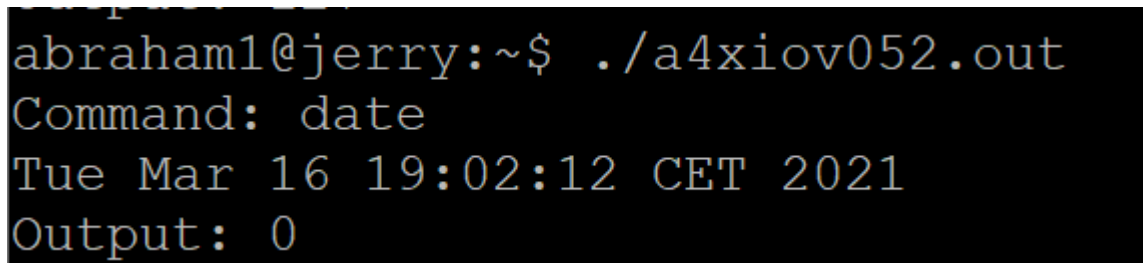
int main(){
    char input[20];
    printf("Command: ");
    scanf("%s", input);
    int status = system(input);

    if(WIFEXITED(status)){
        printf("Output: %d\n", WEXITSTATUS(status));
    }
    else{
        printf("Hamis\n");
    }

    return 0;
}
```

[Read 21 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page



```
abraham1@jerry:~$ ./a4xiov052.out
Command: date
Tue Mar 16 19:02:12 CET 2021
Output: 0
```

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)! Mentés: `parent.c`, ill. `child.c`

```
jerry.iit.uni-miskolc.hu - PuTTY
GNU nano 2.7.4 File: parent.c

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

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

    if(waitpid(pid, NULL, 0)<0)
        perror("wait error");

    return 0;
}
```

[Read 21 lines]

Get Help Write Out Where Is Cut Text Justify Cur Pos Prev Page
Exit Read File Replace Uncut Text To Spell Go To Line Next Page

```
jerry.iit.uni-miskolc.hu - PuTTY
GNU nano 2.7.4 File: child.c

#include <stdio.h>
#include <stdlib.h>

int main(){
    for(int i=0; i<5; i++){
        printf("Ábrahám Péter, A4XIOV\n");
    }

    return 0;
}
```

```
abraham1@jerry:~$ ./parent.out
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
```

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! Mentés: *neptunkodgyak4.c*

```
GNU nano 2.7.4 File: a4xiov054.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main(){
    int pid;
    int status;

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

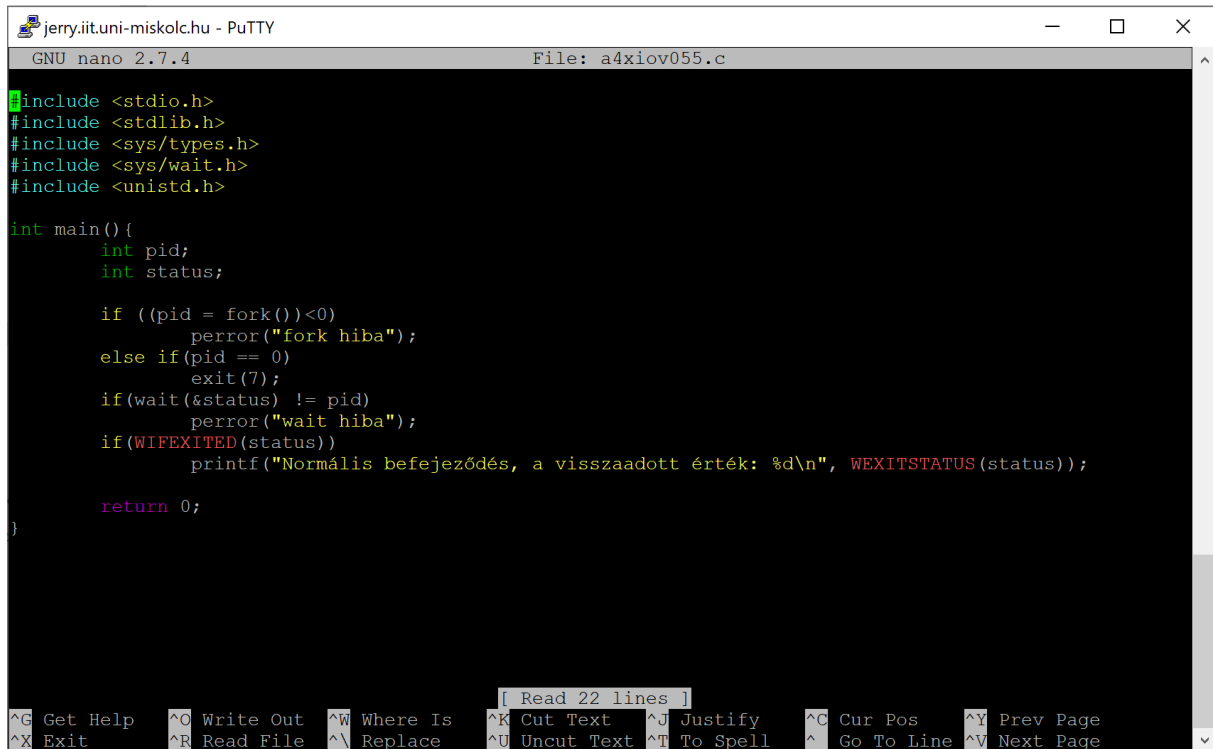
    return 0;
}
```

[Read 20 lines]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos ^Y Prev Page
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line ^V Next Page

```
abraham1@jerry:~$ ./a4xiov054.out
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
Ábrahám Péter, A4XIOV
```

5. A `fork()` rendszerhívással hozzon létre gyerekeket, várja meg és vizsgálja a befejeződési állapotokat (gyerekekben: `exit`, `abort`, nullával való osztás)! Mentés: *neptunkodgyak5.c*



```
GNU nano 2.7.4 File: a4xiov055.c

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

int main() {
    int pid;
    int status;

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

    return 0;
}
```

```
abraham1@jerry:~$ ./a4xiov055.out
Normális befejeződés, a visszaadott érték: 7
```