Operációs rendszerek BSc

5. Gyak.

2022. 03. 07.

Készítette:
Hauer Attila Árpád Bsc
Szak Mérnökinformatikus
Neptunkód JJL4WE

Miskolc, 2022

1. A system() rendszerhívással hajtson végre létező és nem létező parancsot, és vizsgálja a visszatérési érteket, magyarázza egy-egy mondattal A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba. Mentés: neptunkod1fel.c

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

int main()
{
    int status;

    if((status = system("date")) < 0)
        perror("system() error");

    if(WIFEXITED(status))
        printf("Normalis befejezodes, visszaadott ertek = %d\n", WIFEXITED(status));

}
```

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-\) - magyarázza egy-egy mondattal A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba. Mentés: neptunkod2fel.c

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

int main()
{
    char input[100];
    printf("Adjon meg egy parancsot: ");
    scanf("%s", input);
    system(input);

    return 0;
}
```

```
Adjon meg egy parancsot: uname
Linux
```

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 (10-ször) (pl. a hallgató neve és a neptunkód)! - magyarázza egy-egy mondattal A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba. Mentés: parent.c, ill. child.c

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

int main()
{
    for(int i = 0; i < 10; i++)
    {
        printf("Hauer Attila Arpad, JJL4WE\n")
        sleep(2);
    }

    return 0;
}
```

Xy parent.c

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

int main()
{
    pid_t 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;
}
```

xy child.c

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! - magyarázza egy-egy mondattal. A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba. Mentés: neptunkod4fel.c

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

int main()
{
    pid_t pid;

    if((pid = fork()) < 0)
    {
        perror("fork error");
    }
    else if(pid == 0)
    {

        if(execlp("ls", "-l", "/home/zsatesz/OS_GYAK", NULL) < 0)
            perror("execl error");
    }
    if(waitpid(pid, NULL, 0) < 0)
    {
        perror("wait error");
    }

    return 0;
}
```

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

egy-egy mondattal! A fordítás/futtatás után készítsen egy képernyőképet (minden parancs esetén) és illessze be a dokumentumba. Mentés: neptunkod5fel.c

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

int main()
{
    pid_t pid, status;

    if((pid = fork()) < 0)
    {
        perror("Hiba a forkban!");
        exit(7);
    }
    else if(pid == 0)
    {
        abort();
        if(wait(&status) != pid)
        {
            perror("Hiba a wait-ell!");
        }
    }
    if(WIFEXITED(status))
    {
        printf("Sikeres!");
    }

    return 0;
}
```