Operációs rendszerek – 5. Gyakorlat

Linux OS - Rendszerhívások

Töltse fel az aktuális mappába: **Neptunkod_....**

Jegyzőkönyv neve: *neptunkodgyak5.pdf*

Forrás file-ok
fordítás: cc forras.c -o futtathato.out

Feladatok

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

#include <stdlib.h>

int system(const char *command);

#include <sys/types.h>

#include <sys/wait.h>

WIFEXITED(status) igazat ad, ha exit(), _exit() vagy return() volt a befejezés

WEXITSTATUS(status) makró a visszatérési értéket adja (szignifikáns részét)

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*

3. Készítsen egy `parent.c` és egy `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*

#include <unistd.h>
pid_t fork(void);

#include <sys/types.h>
#include <sys/wait.h>
*pid_t wait(int *wstatus);*
*pid_t waitpid(pid_t pid, int *wstatus, int options);*

#include <unistd.h>
*int execl(const char *path, const char *arg, ...*
/ (char *) NULL */);*
*int execlp(const char *file, const char *arg, ...*
/ (char *) NULL */);*

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

4. A `fork()` rendszerhívással hozzon létre egy gyerek processzt-t és abban hívjon meg egy `exec` családbeli rendszerhívást (pl. `execlp`) egy unix-paranccsal. A szülő várja meg a gyerek futását! Mentés: *neptunkodgyak4.c*

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

```
#include <stdlib.h>
void exit(int status);

#include <stdlib.h>
void abort(void);
```

#include <sys/types.h>

#include <sys/wait.h>

WIFEXITED(status) igazat ad, ha exit(), _exit() vagy return() volt a befejezés

WEXITSTATUS(status) makró a visszatérési értéket adja (szignifikáns részét)

WIFSIGNALED(status) igazat ad vissza, ha el nem kapott szignál okozta a terminálódást

WTERMSIG(status) makró az abortálást okozó szignál azonosítóját adja

```
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));
```

```
if ( (pid = fork()) < 0)  perror("fork hiba");
        else if (pid == 0)        /* gyermek */
                abort();          /* befejeződik   */
if (wait(&status) != pid) perror("wait hiba"); /* szülő */
if (WIFSIGNALED(status))
        printf("Abnormális befejeződés, a szignál sorszmáma = %d\n", WTERMSIG(status));
```