

ZOS cv 7/2012

L. Pešička

A decorative graphic element consisting of several horizontal lines of varying lengths and colors (teal and white) extending from the right side of the slide.

Pojmenovaná roura

- `cd /tmp/pesi`
- **`mkfifo roura`** {nebo **`mknod roura p`**}
- `ls -l ; file roura`

- `prw----- 1 pesicka users 0 2010-11-10 05:46 roura`

1. Terminál

- `cat /etc/passwd > roura`

2. Terminál

- `cat /tmp/pesi/roura`

- `rm roura`

Prompt 1. terminálu se znovu neobjeví, dokud nepřčteme data z roury,
Stejně tak pokud napřed pustíme druhý terminál, blokuje, dokud nejsou data v rouře

Alias

- alias pozdrav='echo nazdarek'
 - Vytvoření aliasu
- pozdrav
 - Použití aliasu
- alias
 - Výpis aliasu
- unalias pozdrav
 - Zrušení aliasu
- alias ll='ls -l'

Procesy

Vytvoření nového procesu v Linuxu (Unixu)
systémovým voláním **fork()**

```
int id;  
id = fork();  
if (id == 0)  
    printf(„jsem potomek\n“);  
else  
    printf(„jsem rodič, potomek má PID %d\n“, id);
```

Zde běží 1 proces

Zde běží 2 procesy
liší se návratovou
hodnotou forku, tj.
proměnnou id

Celý příklad - fork1.c

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

```
int main (void) {
```

```
int i;
```

```
i = fork();
```

```
if (i == 0)
```

```
    printf ("Jsem potomek s pidem %d, rodic ma %d\n", getpid(),  
getppid());
```

```
else
```

```
    printf ("Jsem rodic s pidem %d, potomek ma %d\n", getpid(), i);  
}
```

Otestování pod Linuxem

- uložte předchozí příklad jako `fork1.c`
- přeložte: **`gcc -o fork1 fork1.c`**
- spusťte: **`./fork1`**

úkol pro vás:

modifikujte příklad na zombii

zombie: potomek skončí, ale rodič ne a nepřechte si jeho návratový kód

Zombie - možná modifikace

```
#include <stdio.h>

int main (void) {

int i,j;

i = fork();

if (i == 0)
    printf ("Jsem potomek s pidem %d, rodic ma %d\n", getpid(), getppid());
else {

    printf ("Jsem rodic s pidem %d, potomek ma %d\n", getpid(), i);
    for (j=10; j<100; j++) j=11;
}
}
```

Úkoly s forkem()

Ověřte programem

Máme fragment kódu, nakreslete strom procesů a určete, kolikrát se vypíše příslušný řetězec:

```
fork();  
fork();  
printf (“prvni priklad\n”);
```


Další příklady

```
if (fork() == 0)
```

```
    fork();
```

```
    “printf(“druhy prikklad\n”);
```

```
----- další příklad -----
```

```
int i;
```

```
for (i=0; i<2; i++)
```

```
    fork();
```

```
printf(“treti prikklad\n”)
```

Základní operace s procesy

Systemové volání	Funkce
fork()	Vytvoření procesu
wait()	Čekání na dokončení procesu
exit()	Ukončení procesu
execl()	Nahradí aktuální kód procesu kódem z daného souboru Celá rodina volání exec*

Úkol

Napište program f4.c:

- Rodič čeká na dokončení potomka `wait()`
- Potomek vypíše `jsem potomek`
- Potomek spustí program `/bin/date`

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

```
int main(void) {
    int id;
```

```
    id = fork();
```

```
    if (id == 0) {
        printf("jsem potomek\n");
        execl("/bin/date", "date" NULL);
    }
```

```
    else {
        printf("jsem rodic, potomek ma PID %d\n",id);
        wait();
    } }
```

Další možnost na některých
platformách:
execl("/bin/date", NULL)

úkol:
dejte za execl
printf("Sem uz nedojdu");
zkuste spustit neexistující
program

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

int main(void) {

int id, id2;

id = fork();
if (id == 0) {
    printf("jsem potomek\n");
    sleep(5);
} else {
    id2 = fork();
    if (id2 == 0 ) {
        printf("jsem druhy potomek\n");
        sleep(5);
    }
else {
    wait(); wait(); printf("oba potomoci skončili\n");}
}
}
```

Nakreslete graf,
vyjadřující posloupnost
procesů

Procesy

- po vytvoření forkem jsou procesy zcela samostatné
- chceme-li aby „něco“ sdíleli, musíme to naopak explicitně zařídit
- Podobný problém není u vláken
 - Vlákna sdílejí globální proměnné (ale zas řešíme souběžný přístup – kritická sekce)

Ukázky

- fork.zip: fork
 - Ukázka, že proměnné nejsou sdílené
- fork.zip: fork_exec
 - Ukázka spuštění jiného programu včetně ošetření chybových stavů

Ukázkové příklady najdete na portále v courseware v sekci
ZOS - Cvičení – C,Java příklady
(díky přičinění pana kolegy P. Bžocha)

Ukázky IPC

InterProcess Communication

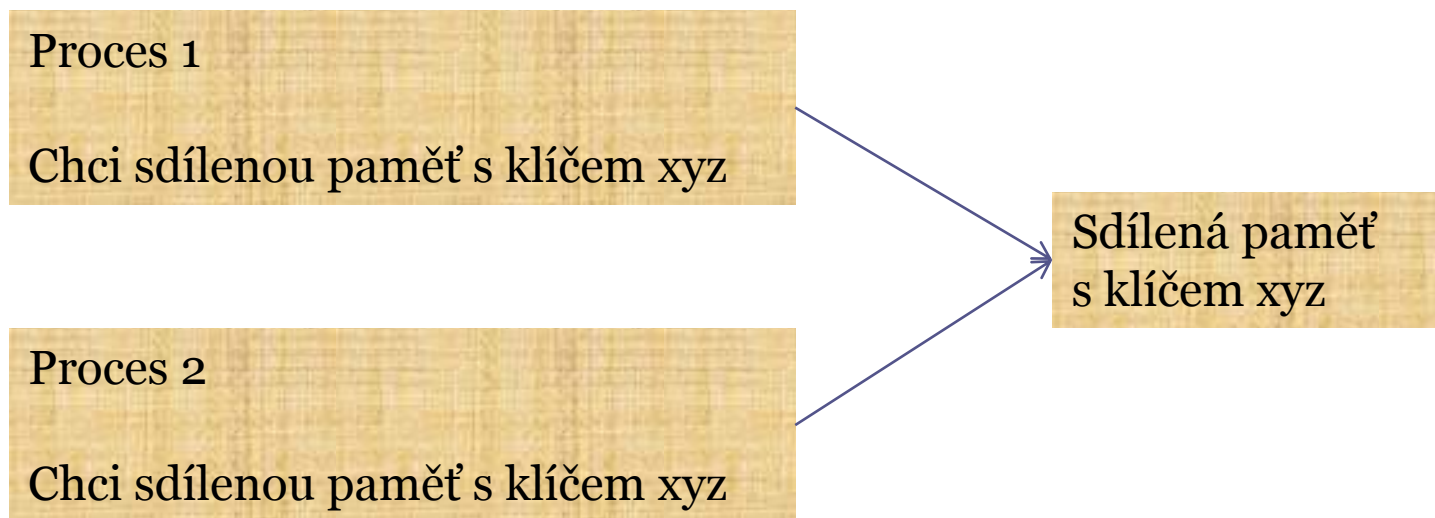
- Prostředky komunikace mezi procesy

Ukazují použití

- Posílání zpráv mezi procesy
- Použití nepojmenované roury mezi příbuznými procesy
- Použití sdílené paměti

Ukázky IPC

Sdílená paměť



Sdílená paměť

- **shmget()** .. vytvoření sdíleného paměťového segmentu
- **shmctl()** .. vlastnictví jinému uživateli, nastavení..
- **shmat()** .. připojení sdíleného segmentu do paměťového prostoru procesu (shmdt odpoj)

poznámka:

dalším používaným způsobem je mmap()

Nepojmenovaná roura

```
#include <unistd.h>  
int sharedPipe[2];  
pipe(sharedPipe);
```

- vytvoří rouru
 - `sharedPipe[0]` .. read end
 - `sharedPipe[1]` .. write end

Zápis procesů cobegin - coend

- Využití maximálního paralelismu
 - Spustit vše, co je v daný okamžik možné

- Př:

Stavíme kolejiště. Nejprve můžeme současně začít kupovat koleje (p1), shánět mašinky (p2) a stavět podkladovou desku (p3). Jakmile máme koupené koleje a postavenou podkladovou desku, můžeme začít stavět koleje (p4). Když jsou koleje postavené a mašinky koupené, vypravíme vlak (p5). Zprovoznění kolejiště následně oslavíme jahodovým džusem (p6).