Katedra počítačov a informatiky
Fakulta elektrotechniky a informatiky
Technická univerzita v Košiciach

Operačné systémy
Zbierka úloh zo systémového programovania

(verzia 1)

Práca so súbormi I – úlohy na samostatnú prácu

1. Overte a zdokumentujte (textové kópie obrazoviek, alebo presmerovanie výstupu), ako sa prejaví použitie resp. nepoužitie príznaku O_CREAT pri otvorení neexistujúceho resp. už existujúceho súboru. Dokumentujte návratovú hodnotu služby a hodnotu premennej **errno**, prípadne výpis funkcie perror(3) – viď *man 3 perror***.**
2. Overte a zdokumentujte (textové kópie obrazoviek, alebo presmerovanie výstupu), ako sa prejaví použitie príznakov O_CREAT|O_EXCL pri otvorení neexistujúceho resp. už existujúceho súboru. Dokumentujte návratovú hodnotu služby a hodnotu premennej **errno**, prípadne výpis funkcie perror(3) – viď *man 3 perror***.**
3. Vyskúšajte a zdokumentujte (textové kópie obrazoviek, alebo presmerovanie výstupu), ako sa prejaví na návratovej hodnote služby **read()** čítanie väčšieho počtu bajtov ako je k dispozícii (napr. máme súbor dĺžky 7, read() sa snaží prečítať, povedzme, 10 bajtov). Dokumentujte návratovú hodnotu služby a prípadne aj hodnotu premennej **errno**, prípadne výpis funkcie perror(3) – viď *man 3 perror***.**
4. Vyskúšajte a zdokumentujte (textové kópie obrazoviek, alebo presmerovanie výstupu), ako sa prejaví na návratovej hodnote služby **read()** pokus o čítanie obsahu súboru, ak sa súborový kurzor nachádza za posledným bajtom súboru. Dokumentujte návratovú hodnotu služby a prípadne aj hodnotu premennej **errno**, prípadne výpis funkcie perror(3) – viď *man 3 perror***.**
5. Vytvorte program, ktorý bude kopírovať obsah súboru zo štandardného vstupu na štandardný výstup až po koniec vstupného súboru. Kopírujte po jednom znaku/bajte. Vyskúšajte funkčnosť programu na kopírovaní súborov pomocou presmerovania štandardného vstupu a výstupu.
6. Overte a zdokumentujte (textové kópie obrazoviek, alebo presmerovanie výstupu), ako sa prejaví na návratovej hodnote služby **lseek()** presun súborového kurzora o niekoľko bajtov v oboch smeroch vzhľadom na:
    a. jeho aktuálnu pozíciu,
    b. začiatok súboru
    c. koniec súboru

    Dokumentujte návratovú hodnotu služby a prípadne aj hodnotu premennej **errno** , prípadne výpis funkcie perror(3) – viď *man 3 perror***.**

7. Napíšte, overte funkčnosť a zdokumentujte program, ktorý v súbore (napr. obsahujúcom iba malé znaky, cca 30 bajtov/znakov) prepíše 3 znaky od pozície 10 znakmi „A“. (obsah súboru vypíšte napr. príkazom **cat(1)** – viď *man 1 cat*)
8. Napíšte, overte funkčnosť a zdokumentujte program, ktorý bude funkčne ekvivalentný programu z bodu 11 (predchádzajúci príklad), avšak počet znakov, pozíciu a znak, ktorým sa má obsah prepisovať, načítajte z povelového riadku ako parametre spúšťaného súboru.

Práca so súbormi II, adresáre – úlohy na samostatnú prácu

1. Overte/preukážte, či po zduplikovaní deskriptora súboru službou `dup()`/`dup2()` sa zapamätá pozícia kurzora aj v alternatívnom kanáli.
2. Pomocou služby `stat()` zrealizujte výpis informácií o súbore vo formáte podobnom výpisu príkazom „`ls -l`“
3. Vytvorte vlastný príkaz `ln`, napr. tak, že bude akceptovať opačné poradie existujúceho a vytváraného súboru.
4. Vytvorte program, ktorý pre zadané meno súboru vypíše, či sa jedná o súbor alebo adresár. Úlohu riešte pomocou funkcií (man 3) pre prácu s adresármi.
5. Vytvorte program, ktorý pre zadané meno súboru vypíše číslo i-uzlu súboru. Úlohu riešte pomocou funkcií (man 3) pre prácu s adresármi.

Ovládanie zariadení – úlohy na samostatnú prácu

1. Zistite a opíšte, aké ďalšie možnosti v nastavení prístupových práv existujú okrem klasických „rwx“ práv.
2. Predefinujte znak pre prerušenie programu – zvyčajne CTRL/C – samozrejme, prostredníctvom programu.
3. Napíšte segment programu pre načítanie loginu a hesla. Pred čítaním hesla vypnite echovanie znakov a po načítaní ho následne opäť zapnite. Následne načítane heslo vypíšte. Pre ovládanie echa použite relevantnú službu jadra resp. vhodné funkcie.
4. Prepnite terminál do režimu, v ktorom jadro OS nebude bufrovať znaky (stačí príkazom stty, ale môžete aj programovo). To znamená, že napr. znaky načítané cez getchar() a následne vypisované z programu na obrazovku (napr. programom - viď Obr.1) budú zobrazované okamžite znak po znaku a nie až po zadaní ENTER (POZOR! – nejedná sa o echované znaky, ale znaky, vypisované z programu; odporúčame vypnúť echo; ak echo nevypnete, tak napr. pri zadaní vstupu „abc<ENTER>“ výstup bude v tvare “aabbcc…” a nie “abc<novy riadok>abc”).

```
#include <stdio.h>

main()
{ int c;
  while((c=getchar())!=EOF)
    putchar(c);
}
```

Obr. 1

Procesy – úlohy na samostatnú prácu

1. Vytvorte program, ktorý spustí nejaký iný program (bez vytvárania nového procesu). Zdokumentujte process ID a parent process ID pôvodného aj novospúšťaného programu. Vyvoďte závery.
2. Vyskúšajte odovzdať dáta z jedného programu do druhého programu (simulujte odovzdanie argumentov z povelového riadku do spúšťaného programu).
3. Vytvorte program, ktorý vytvorí nový proces. Zdokumentujte PID a PPID jednotlivých procesov.
4. Overte, či v dcérskom procese, hneď po jeho vytvorení, sa zachovávajú hodnoty premenných tak, ako boli nastavené v čase vytvorenia nového procesu v rodičovskom procese.
5. Zistite, či zmena hodnoty premennej v dcérskom procese spôsobí (alebo nespôsobí) zmenu zodpovedajúcej premennej v rodičovskom procese.
6. Program vytvoril kanál na zápis do súboru. Nasledovalo vytvorenie nového procesu a potom oba procesy začali do tohto súboru zapisovať. Identifikujte/označte zápisy od jednotlivých procesov (napr. cez PID prefix). Analyzujte vytvorený súbor z hľadiska poradia zapisovania. Program spusťte viackrát a porovnajte výsledky.
7. Vytvorte jednoduchý interpretátor príkazov (shell), ktorého jedinou úlohou bude spúšťať ľubovoľné programy bez parametrov (napr. ls, pwd, ...) – opakovane, ako bash. Príkaz bude zadaný na štandardnom vstupe s absolútnym menom súboru.
8. Vytvorte jednoduchý interpretátor príkazov (shell), ktorého úlohou bude opakovane (viackrát) spúšťať ľubovoľné programy s parametrami (napr. cat <súbor>, ...). Okrem toho bude poznať jeden interný príkaz CD <adresár>. Príkaz bude zadaný na štandardnom vstupe s absolútnym menom súboru/adresára.

Pipe a signály – úlohy na samostatnú prácu

Pipe-y (nepomenované)

1. Vytvorte nepomenovaný pipe, zapíšte doň blok dát (ľubovoľne veľký) a prečítajte ho naspäť.
2. Vytvorte nepomenovaný pipe. Zapíšte doň blok dát. V tom istom procese spusťte iný program (služba `execve()`), ktorý zapísané dáta prečíta naspäť a vypíše ich na konzolu. Informáciu o relevantnom deskriptore súboru odovzdajte pri spúšťaní druhého programu.
3. Prihláste sa do systému trikrát (napr. 3x spustené putty). V každom terminálovom okne zistite názov špeciálneho súboru, ktorý ho reprezentuje (príkaz `tty`). Vytvorte program, ktorý vytvorí nepomenovaný pipe a následne vytvorí (fork-ne) dva nové procesy - potomky aktuálneho procesu. Následne, pôvodný proces zapíše (resp. opakovane zapisuje) do pipe-u blok dát (napr. „abcdefghijklm“). Dva novovytvorené procesy budú načítavať z pipe-u *po jednom bajte* a vypisovať načítané znaky do zvyšných dvoch terminálových okien (POZOR!!! nie STDOUT, ale otvoriť špeciálny súbor získaný príkazom `tty` a zapisovať cez zodpovedajúci file deskriptor). Sledujte, zdokumentujte a vysvetlite distribúciu (vypisovaných) dát medzi procesmi.
4. Prihláste sa do systému najmenej trikrát (napr. 3x resp. viackrát spustené putty). V každom terminálovom okne zistite názov špeciálneho súboru, ktorý ho reprezentuje (príkaz `tty`). Vytvorte program, ktorý vytvorí nepomenovaný pipe a následne vytvorí (fork-ne) korešpondujúci počet nových procesov - potomky aktuálneho procesu (potiaľ to zodpovedá úlohe 3). Následne, jeden program začne na STDOUT a zároveň do pipe-u zapisovať správy premenlivej dĺžky vo formáte (STDOUT textovo, pipe binárne):

   ```
   <sprava> ::= <hlavicka><telo>
   <hlavicka> ::= dlzka_typ_cele_cislo_32_bitov
   <telo> ::= {a|b|c|d|…|z}+
   ```

   teda správa obsahuje hlavičku určujúcu dĺžku správy a ľubovoľný počet ľubovoľných znakov `a-z`, v počte jeden alebo viac (použite napr. generátor náhodných čísel). Správy odosielajte vo fixných 1-3 sekundových intervaloch; stačí poslať 5-10 správ; každú správu najprv sformujte celú u odosielateľa a potom ju odošlite jedným volaním korešpondujúcej služby. Ostatné programy striedavo prijímajú tieto správy (najprv hlavičku, na základe hlavičky – dĺžky správy - zvyšok správy) a vypisujú ich na obrazovku. Zdokumentujte (kópie obrazoviek, vysvetlenie) správanie programového systému, identifikujte (ale neriešte) prípadne problémy fungovania tohto systému.

   Poznámka: stačí, ak programový kód jednotlivých procesov umiestnite do zodpovedajúcich sekcií jedného a toho istého zdrojového kódu. Vyhnete sa tým „problémom“ s odovzdávaním identifikátorov pre jednotlivé konce pipe-u.

Signály

5. Vytvorte program, ktorý v rámci „hlavnej postupnosti výpočtu“ uspíte na cca 10-20 sekúnd. Zároveň zabezpečte, aby tento program, po prijatí signálu SIGUSR1, vypísal na terminál správu „Prijal som signál SIGUSR1“. Následne tento program spusťte na pozadí (znak „&“ na konci príkazového riadku) a začnite mu posielať signál SIGUSR1 (stačí napr. príkazom `kill`). Zdokumentujte správanie systému – kópie obrazoviek, vysvetlite správanie.

Pipe-y a signály spolu

6. Podobne, ako v úlohe 3, v časti pipe-y, vytvorte programový systém naviazaný na tri rôzne terminálové okná s tým rozdielom, že dcérske procesy budú vypisovať dáta (jednorázovo - znak, prípadne celý riadok) len vtedy, keď dostanú na to požiadavku cez signál SIGUSR2. Interval medzi jednotlivými odoslaniami signálov – minimálne 2 sekundy.
7. Podobne, ako v úlohe 5, v časti pipe-y (nepomenované), vytvorte programový systém naviazaný na tri rôzne terminálové okná s tým rozdielom, že dcérske procesy budú vypisovať prijaté správy len vtedy, keď dostanú na to požiadavku cez signál SIGUSR2. Keďže správy sa odosielajú v 1-3 sekundových intervaloch, dodržte tento interval aj pri posielaní signálov (napr. pošlete signál po odoslaní správy).

Zdieľaná pamäť – úlohy na samostatnú prácu

1. Vytvorte programom v jazyku C/C++ segment zdieľanej pamäte. Zdokumentujte vytvorený segment zdieľanej pamäte príkazom „`ipcs -m`“ – výpisom zoznamu segmentov pred operáciou a po nej. Zamerajte sa na identifikátor zdieľanej pamäte, vlastníka a prístupové práva.
2. Vytvorte program, ktorým zmažete segment zdieľanej pamäte vytvorený v bode 1. Zdokumentujte príkazom „`ipcs -m`“ pred zmazaním a po zmazaní.
3. Vytvorte segment zdieľanej pamäte s prístupom pre - skupinu a ostatných - iba na čítanie. Zdokumentujte vytvorený segment zdieľanej pamäte príkazom „`ipcs -m`“. Zamerajte sa predovšetkým na prístupové práva.
4. Vytvorte dva programy - P1 a P2. Program P1 vytvorí segment zdieľanej pamäte a zapíše doň ľubovoľný textový reťazec. Program P2 sa pripojí k tomu istému segmentu zdieľanej pamäte a vypíše reťazec, ktorý tam je zapísaný. Spusťte program P1 a po jeho ukončení spusťte program P2. (Cieľom je vypísať pomocou programu P2 textový reťazec, ktorý tam zapísal program P1.) Zdokumentujte.

Synchronizácia / semafóry – úlohy na samostatnú prácu

1. Vytvorte dva programy - P1 a P2. Program P1 vytvorí segment zdieľanej pamäte a v cykle bude doň zapisovať textový reťazec zadávaný na štandardnom vstupe. Program P2 sa pripojí k tomu istému segmentu zdieľanej pamäte a bude vypisovať reťazce, ktoré tam zapísal program P1. Spusťte program P1 v jednom okne (putty, xterm, ...) a po ňom program P2 v druhom okne (putty, xterm, ...). Následne zadávajte ľubovoľné reťazce v okne, kde beží P1 a pozorujte okno, kde beží P2. Cieľom je, aby program P2 vypisoval reťazce zadávané programu P1. Pokúste sa zabezpečiť synchronizáciu procesov P1 a P2 (teda, aby proces P2 vypisoval každý reťazec, ktorý proces P1 zapíše do zdieľanej pamäte, práva raz) špeciálnym príznakom v zdieľanej pamäti (príznak, ktorý bude indikovať, či reťazec v zdieľanej pamäti je platný alebo nie)
2. Vytvorte dva programy - P1 a P2. Program P1 vytvorí segment zdieľanej pamäte a v cykle bude doň zapisovať textový reťazec zadávaný na štandardnom vstupe. Program P2 sa pripojí k tomu istému segmentu zdieľanej pamäte a bude vypisovať reťazce, ktoré tam zapísal program P1. Spusťte program P1 v jednom okne (putty, xterm, ...) a po ňom program P2 v druhom okne (putty, xterm, ...). Následne zadávajte ľubovoľné reťazce v okne, kde beží P1 a pozorujte okno, kde beží P2. Cieľom je, aby program P2 vypisoval reťazce zadávané programu P1. Zabezpečte synchronizáciu procesov P1 a P2 (teda, aby proces P2 vypisoval každý reťazec, ktorý proces P1 zapíše do zdieľanej pamäte, iba raz) pomocou existencie/neexistencie súboru v súborovom systéme (synchronizácia službou open()).
3. Vytvorte dva programy - P1 a P2. Program P1 vytvorí segment zdieľanej pamäte a v cykle bude doň zapisovať textový reťazec zadávaný na štandardnom vstupe. Program P2 sa pripojí k tomu istému segmentu zdieľanej pamäte a bude vypisovať reťazce, ktoré tam zapísal program P1. Spusťte program P1 v jednom okne (putty, xterm, ...) a po ňom program P2 v druhom okne (putty, xterm, ...). Následne zadávajte ľubovoľné reťazce v okne, kde beží P1 a pozorujte okno, kde beží P2. Cieľom je, aby program P2 vypisoval reťazce zadávané programu P1. Zabezpečte synchronizáciu procesov P1 a P2 (teda, aby proces P2 vypisoval každý reťazec, ktorý proces P1 zapíše do zdieľanej pamäte, iba raz) pomocou semaforu.

Networking – úlohy na samostatnú prácu

1. Vytvorte dva programy - P1 a P2. Program P1 vytvorí server (deamon) čakajúci na správu od akéhokoľvek klienta (textový reťazec). Po prijatí server prijatý reťazec vypíše na štandardný výstup, počká zopár sekúnd (náhodne, povedzme 1-3 sekundy; sleep(), random()) a následne pošle správu „VYPISANE!“. Program P2 – klient - načíta zo štandardného vstupu reťazec a pošle ho serveru (deamonu) vytvorenému programom P1. Po prijatí potvrdenia od procesu P1, toto potvrdenie vypíše. Na komunikáciu použite protokol TCP/IP, číslo portu zvoľte ľubovoľne ale zmysluplne. Spusťte program P1 v jednom okne (putty, xterm, ...) a po ňom program P2 v druhom okne (putty, xterm, ...), pričom oba programy pobežia na tom istom počítači. Následne zadávajte ľubovoľné reťazce v okne kde beží P2 a pozorujte okno, kde beží P1.
2. Modifikujte úlohu 1 tak, že namiesto protokolu TCP/IP použijete protokol UDP/IP.
3. Modifikujte úlohu 1 tak, že nebudete vypisovať jednoduchý reťazec, ale vzdialene vypíšete obsah celého súboru, názov ktorého zadáte na štandardnom vstupe alebo ako parameter povelového riadku klienta.