UNIVERZA V LJUBLJANI

Fakulteta za elektrotehniko

Vzporedni sistemi – gradivo za laboratorijske vaje 2005/2006

janez.pers@fe.uni-lj.si

1. UVOD

Pri vajah pri predmetu Vzporedni sistemi (VS) boste v praksi preizkusili nekaj sistemskih klicev in orodij, ki so namenjena podpori izvajanju paralelnih algoritmov pod operacijskim sistemom UNIX (fork in wait, cevi, sporočila, vtičnice, deljen pomnilnik, semaforji). Razvili boste eno ali več rešitev s področja vzporednega procesiranja, pri tem pa uporabili znanje, ki ste ga dobili na predavanjih. V tem tekstu so opisana nekatera opravila (urejanje teksta, prevajanje...) v kontekstu programske opreme, ki je trenutno nameščena na računalnikih, na katerih bodo tekle vaje.

Ta navodila z vajami in še nekatera gradiva za predmet lahko dobite na internetnem naslovu: http://vision.fe.uni-lj.si/classes/VS

2. POTEK VAJ IN OPREMA

Vaje bodo tekle na osebnih računalnikih z operacijskim sistemom Linux v učilnici Laboratorija za slikovne tehnologije (LST – v 4. etaži stare stavbe). V terminu, ki je vajam namenjen, bo v učilnici prisoten asistent.

Na vsakem od računalnikov je poleg Windows XP nameščen še Linux, in sicer distribucija RedHat 9.

2.1. Start Linuxa

Če na računalniku, na katerem nameravate delati tečejo Windows XP, jih najprej zaustavite (shutdown/restart) in počakajte da se računalnik restarta. Ko se prikaže na ekranu NT Boot Loader, imate nekaj sekund časa, da izberete »Linux«, sicer se bodo zagnali Windows XP. Zagon Linuxa lahko traja do nekaj minut - počakajte, dokler se računalnik ne preklopi v grafični način in se prikaže dialog za prijavo. Preden prižigate računalnik, preverite, ali ni že morda prižgan, in je treba le vklopiti monitor ali premakniti miško!

2.2 Start računalniške gruče (cluster)

Nekatere vaje bodo zahtevale izvedbo rešitev v vzporedni izvedbi, ki bo delovala na večih računalnikih naenkrat. V tem primeru morate počakati, da se zagon računalnikov, ki jih nameravate uporabljati, začasno zaustavi s sporočilom o povezavi v gručo. Postopek zagona Linuxa nadaljujte po navodilih, ki se takrat izpišejo.

2.2. Prijava v sistem

Vpišite uporabniško ime in geslo. Geslo je na začetku enako uporabniškemu imenu, če pa ga hočete spremeniti, morate spremembo izvesti na vsakem računalniku posebej z ukazom `passwd` iz ukazne vrstice.

2.3. Zaustavitev Linuxa

Računalnika NE resetirajte s pomočjo tipke na ohišju in ga NE ugasnite, ne da bi prej zaustavili sistem! Najlažje boste Linux pripravili na izklop tako, da se v grafičnem vmesniku odjavite (logout), potem pa izberete opcijo »shutdown«. Pomembno: če je vaš računalnik del gruče, preverite, ali so delo na njem zaključili tudi ostali!

3. RAZPOREDITEV V SKUPINE

Laboratorijske vaje se bodo izvajale v skupinah s po dvema študentoma. Vsakemu paru pripada eno uporabniško ime. Računalniki so sicer povezani v LAN omrežje, vendar ima vsak računalnik svoj niz uporabniških direktorijev `/home/vs01, /home/vs02 ..` in nastavitev. Posledica tega je, da so vaše nastavitve (kar velja TUDI za geslo!) omejene na računalnik, na katerem ste nazadnje delali.

3.1. Shranjevanje in prenašanje podatkov

Ker so uporabniški direktoriji posameznih računalnikov ločeni med sabo, **vedno** shranite vse, kar ste tisti dan naredili (to bodo večinoma .c datoteke s programsko kodo) na svojo disketo. Postopek je opisan v nadaljevanju tega teksta. Ne zanašajte se na to, da bodo datoteke ostale na disku tam, kjer ste jih pustili - že zaradi morebitnih okvar diskov. Alternativa temu je, da si rezultate svojega dela pošljete po elektronski pošti – če imate dostop do nje preko spletnega vmesnika.

4. UREJANJE PROGRAMOV IN PREVAJANJE POD OKOLJEM KDE/LINUX

4.1 Ukazna školjka (shell) in terminal

Po prijavi v računalnik se boste znašli v grafičnem okolju KDE. Kljub temu boste za večino operacij potrebovali ukazno vrstico. Do te pridete z izbiro podmenija »System tools« in »Terminal« v glavnem meniju grafičnega vmesnika. Za preklop v tekstni način dela (če se pojavi napaka v grafičnem vmesniku) uporabite tipke CTRL+ALT+F1 do F7.

4.2 Urejanje teksta

Za urejanje programov v jeziku C uporabljajte program KWrite. Poženete ga s klikom na ikono svinčnika desno spodaj, ali pa iz ukazne vrstice z ukazom

```
kwrite ime_datoteke &
```

Znak & pove školjki da naj program požene v ozadju (oziroma naj ne čaka z vrnitvijo prompta, dokler se program ne konča). To obliko ukazov uporabljajte vedno, ko boste iz ukazne vrstice poganjali grafične aplikacije.

V programu KWrite si lahko nastavite poljubno velikost znakov in vklopite označevanje teksta glede na sintakso (menu Options,).

PRIMER: "Hello world" v jeziku C.

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

```
int main()
{
    printf("Hello world! \n");
    return 0;
};
```

Programsko kodo začasno shranjujte v svoj domači direktorij (`/home/uporabniško_ime/`) s pripeto končnico .c, v našem primeru naj bo to hello.c. Vsebino tekstovne datoteke lahko izpišete na ekran iz ukazne vrstice z ukazoma

`cat ime_datoteke` ali `more ime_datoteke`

Nasvet: Do svojega domačega direktorija najhitreje pridete od kjerkoli s pomočjo ukaza `cd` brez parametrov.

4.3 Prevajanje programa

Za prevajanje programov bomo uporabili GNU C prevajalnik, ki ga poženete iz ukazne vrstice na naslednji način.

```
gcc vhodna_datoteka –o izhodna_datoteka
```

V primeru, da je prevajanje uspešno, prevajalnik ne sporoči ničesar. Naš program hello.c prevedemo torej z:

```
gcc hello.c –o hello
```

4.4 Zagon programa

Program zaženemo tako, da navedemo pot do njega in ime izvršljive datoteke. Če je trenutni direktorij v sistemski poti (path) lahko navedemo samo ime. V našem primeru to ne deluje:

```
hello
bash: hello: command not found
```

Zato navedimo tudi pot (trenutni direktorij, ki ga predstavlja pika).

```
./hello
Hello world!
```

4.5 Prevajanje programa, ki uporablja zunanje knjižnice

Popravimo nekoliko naš program hello.c, tako, da dodamo za ukazom `printf` novo vrstico. Ukaz sicer v takšnem programčku ne naredi ničesar, in je naveden le kot primer:

```
bproc_move(-1);
```

Program shranimo, in ponovno prevedimo. Prevajalnik vrne napako:

```
undefined reference to bproc_move
```

Na začetek programa najprej dodamo vrstico, s katero vključimo definicijo te funkcije v program:

```
#include <sys/bproc.h>
```

Program pa moramo prevesti tako, da povezovalniku (linker) povemo, da naj program poveže s knjižnico bproc:

```
gcc hello.c –o hello -lbproc
```

4.6 Prenos datotek na disketo

Za kopiranje datotek na disketo ali z nje uporabite grafično okolje (postopek je podoben kot v okolju Windows) ali pa paket `mtools`. Sestavljen je iz večih orodij, ki ustrezajo ukazom, ki jih poznate iz DOSove ukazne vrstice. Imena so podobna kot pod DOSom, s to razliko, da je pred njimi dodana črka m. Nekaj primerov:

```
mdir a:
mcopy a:hello.c /home/vs01
mcopy ~/* a:
mdel a:*.*
```

4.7 Pregled in upravljanje procesov, ki tečejo na računalniku

Procese, ki trenutno tečejo na računalniku lahko vidite s pomočjo ukazov `ps` in `top.` Pod okoljem KDE poženite grafično, Gnome System Monitor, ki je dostopna tudi preko bližnjice, ukaza `ktop:`

```
ktop &
```

Program vam prikaže seznam procesov. Drevo procesov pa vam prikaže programček, ki ki ga aktivrira ukaz:

```
ptree &
```

Tako lahko vidite relacije med procesi-roditelji in procesi-otroci.

4.8 Podrobnejše informacije

Če vas zanimajo podrobnejše informacije o ukazih in sistemskih funkcijah, ki jih boste uporabljali pri svojem delu, uporabite ukaz `man,` na primer:

```
man mtools
man fork   ali    man 2 fork
man 3 sleep
...
```

5. POTEK VAJ, NALOGE IN ZAGOVOR

Vaje bodo sestavljene iz dveh delov, za pristop k izpitu VS morate opraviti oba!

1. DEL
Prvi del vaj predstavlja reševanje nalog, ki so priložene k tem navodilom. Vsaka od vaj je točkovana z ustreznim številom točk, glede na zahtevnost. Rešiti morate toliko nalog, da nanesejo skupaj 100%. Izbira nalog je poljubna, važen je skupni seštevek točk. Vaje boste reševali v skupinah po dva, časa za reševanje pa imate vse do konca semestra, ko boste morali vaje zagovarjati. Pogoja za uspešno oceno zagovora prvega dela vaj sta dva:

- Programi, ki ste jih napisali morajo delovati, kot to predpisuje naloga (potrebni pogoj)
- Razumeti morate, kako program deluje in to pri zagovoru tudi pokazati (zadostni pogoj).

Zagovori se bodo načeloma začeli po novem letu. Ko boste programe končali in jih preizkusili (priporočljivo je, da jih preizkusite na tistem računalniku, na katerem jih boste zagovarjali!), se dogovorite z asistentom za zagovor, ki bi naj bil načeloma v času vaj po urniku. Zagovor vaj je skupen za oba študenta, ki sta sodelovala pri reševanju izbranih nalog. Vaje lahko zagovarja tudi eden od študentov, vendar se s tem šteje, da je vaje opravil le on. V tem primeru mora drug študent vaje zagovarjati posebej, tudi če so te rezultat skupnega dela.

2. DEL
Drugi del vaj predstavlja paralelizacija sekvenčnega algoritma, ki jo boste izvajali pod vodstvom asistenta. Ker je problem razmeroma zahteven, boste dobili že izdelan sekvenčni algoritem s področja obdelave slik, in ga boste predelali tako, da se bo izvajal hkrati v večih procesih. Za uspešno ocenjen drugi del laboratorijskih vaj iz predmeta VS je potrebna prisotnost na teh vajah, ki bodo predvidoma konec novembra ali v začetku decembra. Točen datum boste izvedeli na predavanjih in na oglasni deski pred laboratorijem.

NALOGE

Naloga 1
(35%)

Napišite program, ki bo omogočal pregledovanje podatkov in spreminjanje podatkov, ki so shranjeni v datoteki. Datoteka vsebuje deset nizov, dolžine 20 ASCII znakov. Program naj uporabniku omogoča naslednje operacije:

- spreminjanje posameznega niza, ki ga uporabnik doseže z vpisom ukaza *n=novi_niz* in
- izpis posameznega niza na terminal, ki ga uporabnik doseže z upisom ukaza *n?*.

Pri tem je *n* številka niza (0÷9), *novi_niz* pa je niz znakov, ki bo nadomestil stari niz.
Ali lahko več uporabnikov hkrati uporablja vaš program za delo z isto datoteko? Ali je možno, da eden izmed uporabnikov prebere napačno vsebino datoteke (polovično zapisani niz)?

Naloga 2
(75%)

Potrebno je omogočiti večjemu številu uporabnikov oz. uporabniških procesov hkraten dostop do preproste baze podatkov. Baza podatkov vsebuje 10 nizov, dolžine 20 ASCII znakov. Vsak uporabnik laho izpiše posamezen niz ali spremeni posamezen niz.
Napišite programa *server* in *client*, ki bosta opravljala opisano nalogo. Program *server* bo opravljal vlogo strežnika, bo le eden in bo hranil v pomnilniku vsebino desetih nizov ter stregel zahtevam programov *client* po branju ali vpisu enega izmed nizov. Programov *client* bo lahko več in bodo vsi lahko hkrati zahtevali usluge (branje ali vpis niza) od programa *server*. Program *client* naj uporabniku omogoča naslednje operacije:

- spreminjanje posameznega niza, ki ga uporabnik doseže z upisom ukaza *n=novi_niz* in
- izpis posameznega niza na terminal, ki ga uporabnik doseže z upisom ukaza *n?*.

Pri tem je *n* številka niza (0÷9), *novi_niz* pa je niz znakov, ki bo nadomestil stari niz.
Komunikacijo med procesi realizirajte s pomočjo sistema sporočil (message) ali s pomočjo poimenovanih cevi (named pipe oz. FIFO). Strežnik naj odpre cev ali vrsto, z imenom (ali ključem), ki je znano vsem uporabnikom. Iz te cevi (vrste) bo strežnik bral zahteve procesov *client*. Vsak proces *client* naj ravno tako odpre cev (vrsto), iz katere bo prebral odgovor strežnika. *Client* naj strežniku poleg zahteve po dostopu do podatkov sporoči tudi ime svoje cevi (vrste), v katero bo strežnik poslal odgovor na zahtevo. (Sistem poštnih nabiralnikov).

Naloga 3
(35%)

Napišite program, ki prestreže signal zahteve za prekinitev izvajanja programa *SIGINT* (2), ki se sproži ob pritisku tipke *<CTRL>/C*, ter signal za prenehanje izvajanja programa *SIGTERM* (15), ki ga pošlje program *kill*. Ob vsakem sprejemu signala *SIGINT* naj se program podvoji (fork) in izpiše naslednje sporočilo: *Proces n je ustvaril proces m*, pri čemer je *n* PID procesa, *m* pa PID novonastalega procesa. Ob vsakem sprejemu signala *SIGTERM* naj proces pošlje isti signal svojem očetu in otrokom, izpiše sporočilo *Konec procesa n* (pri čemer je *n* PID procesa) ter preneha z delovanjem.

Naloga 4a
(65%)

Napišite programa za prepis datoteke tako, da bo prvi program (z imenom *from*) ustvaril deljen pomnilnik (shared memory), bral vhodno datoteko in posredoval podatke drugemu programu s pomočjo deljenega pomnilnika. Drugi program (z imenom *to*) naj se priklopi na deljeni pomnilnik, ustvari izhodno datoteko in prenaša podatke iz deljenega pomnilnika v datoteko. Ime vhodne oz. izhodne datoteke podamo programoma kot parametra ukazne vrstice.
Tako bi prepis datoteke *dat1* v datoteko *dat2* dosegli z ukazoma:

>from dat1
>to dat2

Naloga 4b
(100%)

Napišite programa iz naloge 4a tako, da uporabite:
- deljen pomnilnik
- poimenovane cevi (FIFO)

Primerjajte hitrost delovanja programov.

Naloga 5a
(70%)

Večjemu številu uporabnikov želimo omogočiti medsebojno komunikacijo in sicer tako, da se bo besedilo, ki ga vpiše katerikoli uporabnik, izpisalo na zaslonih vseh uporabnikov. Besedilo naj se izpiše takoj, ko uporabnik, ki ga je vpisal, zaključi vrstico (*<CR>*). Vsak uporabnik naj pri zagonu komunikacijskega programa poda svoje ime kot parameter v ukazni vrstici. Besedilo, ki se izpisuje na zaslonih, naj vedno vsebuje tudi ime uporabnika, ki je besedilo vpisal. Uporabniški zasloni naj izgledajo takole:

ime_uporabnika_1 : sporočilo_1
ime_uporabnika_2 : sporočilo_2
…

Pri vsaki vključitvi novega uporabnika naj se vsem uporabnikom izpiše sporočilo **Novi uporabnik : ime_uporabnika*. Pri izključitvi uporabnika pa se izpiše **Izključitev uporabnika : ime_uporabnika*.
Napišite program, ki bo omogočal opisano komunikacijo (s pomočjo deljenega pomnilnika, sporočil ali FIFO).

Naloga 5b
(100%)

Nalogo 5a nadgradite s pomočjo vtičnic po TCP/IP protokolu tako, tako, da bo omogočena komunikacija med uporabniki večih računalnikov naenkrat. Nalogo rešite tako, da bo za komunikacijo med uporabniki skrbel program *strežnik*, znotraj katerega bo medprocesna komunikacija realizirana s pomočjo deljenega pomnilnika, sporočil ali FIFO, uporabniki pa se nanj povezujejo prek vtičnice s programom *odjemalec*. Programu *odjemalec* naj

uporabniki preko ukazne vrstice podajo dva parametra: svoje ime in IP naslov računalnika na katerem teče strežnik.

Naloga 6
(100%)

Napišite program, ki opravi naslednje funkcije:
Program ustvari deljeni pomnilnik (shared memory) ali se priključi na segment deljenega pomnilnika z vnaprej določenim ključem. V pomnilniku naj bodo zapisana števila procesov (PID), ki so že priključeni na pomnilnik. Proces po priključitvi vpiše v deljeni pomnilnik še svojo PID število, ter na standardni izhod izpiše števila vseh že priključenih procesov. Od tega trenutka naprej naj proces ob vsaki spremembi stanja deljenega pomnilnika (tj. ob priključitvi novega procesa ali pri odjavi priključenega procesa) izpiše, kateri proces se je priključil ali odjavil. Izvajanje procesa preneha, če proces sprejeme signal *SIGINT* ali *SIGTERM*. Predno preneha z izvajanjem, naj proces umakne svoje PID število iz deljenega pomnilnika.

Naloga 7
(50%)

Napišite programa za prenos sporočil med dvema računalnikoma z uporabo komunikacijskih vtičnic, po TCP/IP protokolu. Prvi program naj bo strežnik. Ta naj ustvari vtičnico, počaka, da se drugi program priključi nanjo, nakar bere prispele podatke in jih prepisuje na standardni izhod. Drugi program naj se priključi na vtičnico, bere standardni vhod in prebrane podatke prepisuje na vtičnico.

Naloga 8a
(75%)

Napišite programa za prenos datotek med dvema računalnikoma z uporabo vtičnic, po TCP/IP protokolu. Prvi program naj bo sprejemni *strežnik*, ki bo sprejemal datoteke. Ta naj ustvari vtičnico in počaka, da se drugi program (*oddajnik*) priključi nanjo. Po priključitvi oddajnika naj ustvari novi proces - *sprejemnik*, ki bo sprejel podatke od oddajnika in jih zapisal v datoteko, sam pa počaka na naslednjega oddajnika. Drugi program - oddajnik naj se priključi na vtičnico, pošlje sprejemniku ime datoteke, nakar odpre datoteko, jo bere in pošilja sprejemniku. Sprejemnik naj po sprejemu imena datoteke kreira datoteko. Nato naj bere prispele podatke in jih zapisuje v datoteko. Po opravljenem prenosu datoteke se izvajanje oddajnika in sprejemnika zaključi, sprejemni strežnik pa nadaljuje s čakanjem na nov oddajnik.

Naloga 8b
(100%)

Program iz naloge 8a nadgradite tako, da oddajnik odda poljubno izvršljivo datoteko, sprejemnik pa po prenosu datoteke le-to tudi zažene s pomočjo sistemskega klica `exec`. Izvršljivo datoteko, ki jo boste prenašali naredite sami. Na strani sprejemnika naj ob izvajanju izpiše določen niz, ki bo dokazoval, da je bila uspešno prenesena in izvršena (npr. »Hello world«).

Naloga 9
(50%)

Realizirajte binarni semafor z uporabo deljenega pomnilnika. Semafor preizkusite tako, da ustvarite dva procesa, *p1* in *p2*, ki izpisujeta sporočila na standardni izhod. Sporočila naj se izpišejo v naslednjem zaporedju:

1) proces *p1* izpiše besedilo *p1 : prvo sporočilo*
2) proces *p2* izpiše besedilo *p2 : prvo sporočilo*
3) proces *p1* izpiše besedilo *p1 : drugo sporočilo*

4) proces *p2* izpiše besedilo *p2 : drugo sporočilo*
5) proces *p1* izpiše besedilo *p1 : tretje sporočilo*
6) proces *p2* izpiše besedilo *p2 : tretje sporočilo*

Zaporedje izpisov naj bo neodvisno od zaporedja, po katerem smo zagnali procesa *p1* in *p2*.

Naloga 10
(100%)

Realizirajte programa, ki bosta omogočila uporabniku izvrševanje ukazov na drugem računalniku. Prvi program naj bo strežnik za izvrševanje ukazov, ki teče na računalniku, na katerem želimo izvršiti ukaz. Drugi program naj teče na uporabnikovem računalniku, sprejema uporabnikove ukaze in jih posreduje strežniku ter prikazuje rezultate izvrševanja (standardni izhod procesov, ki so bili zagnani na strani strežnika), ki jih vrne strežnik. Strežnik naj omogoča hkratno delo več uporabnikom. Komunikacijo med programi realizirajte z uporabo komunikacijskih vtičnic (sockets) po TCP/IP protokolu.

Naloga 11a
(50%)

Sestavite program, ki bo med množico naravnih števil iskal praštevila. Program realizirajte tako, da bo naloga iskanja praštevil porazdeljena med dva ali več procesov-otrokov, od katerih bo vsak preiskoval določen blok (interval) naravnih števil. Proces-roditelj pa naj na koncu rezultate (najdena praštevila) zbere in izpiše na standardni izhod. Nalogo realizirajte s pomočjo deljenega pomnilnika in semaforjev. Namig: če želite rešiti tudi nalogo 11c, se izogibajte medprocesni komunikaciji do trenutka, ko le-ta postane nujna (prenos rezultatov)!

Naloga 11b
(80%)

Program iz naloge 11a nadgradite tako, da bo deloval nepretrgoma, dokler ga zunanji uporabnik ne bo prekinil. Ko proces-otrok preko deljenega pomnilnika prenese rezultate roditelju in zaključi z delom, naj roditelj določi nov interval števil za preizkušanje ter zažene nov proces, ki deluje nad novim blokom podatkov, tako, da bosta v vsakem trenutku aktivna vsaj dva procesa-otroka.

Ker je možno kompleksnost iskanja praštevil v določenem bloku izračunati vnaprej, poskusite nalogo realizirati na takšen način, da roditelj dolžino blokov odmerja tako, da vsi otroci porabijo približno enako količino časa za obdelavo enega bloka.

Naloga 11c
(100%)

Program iz naloge 11b priredite tako, da bodo otroci tekli vzporedno na dveh ali več računalnikih. Nalogo boste najelegantneje rešili na gruči računalnikov v učilnici, s pomočjo sistema BPROC in njegove funkcije za migracijo procesov `bproc_move`.

Naloga 12a
(80%)

Sestavite program, ki bo med množico naravnih števil iskal praštevila. Rešitev naj bo v obliki dveh ločenih programov, *strežnika* in *odjemalca*. Strežnik naj ustvari vtičnico (socket) po protokolu TCP/IP in čaka, da se nanjo priklopi eden ali več odjemalcev. Strežnik naj vsakemu od odjemalcev pošlje podatke o bloku (intervalu) števil, med katerimi naj odjemalec išče morebitna praštevila.

Vse računanje in iskanje praštevil naj poteka v odjemalcu, strežnik pa naj le zbira končne rezultate in jih izpisuje na standardni izhod. Zveza med odjemalcem in strežnikom naj ostane

vzpostavljena. Ko odjemalec obdela dodeljen blok števil, naj pošlje strežniku rezultate ter počaka na nov blok rezultatov. Program *odjemalec* naj prebere IP naslov računalnika, na katerem teče strežnik iz ukazne vrstice. Program *strežnik* naj omogoča, da pri reševanju problema sodeluje *n* odjemalcev, pri čemer je *n* načeloma omejen le z možnostmi strojne in programske opreme.

luzv1:$ strežnik
luzv2:$ odjemalec 193.2.71.1
luzv3:$ odjemalec 193.2.71.1
luzv4:$ odjemalec 193.2.71.1
...
itd.

Naloga 12b
(100%)

Program iz naloge 12a priredite tako, da bo strežnik poskušal enakomerno porazdeljevati breme računanja med (različno hitre) odjemalce. Ker je možno kompleksnost iskanja praštevil v določenem bloku izračunati vnaprej, naj strežnik na podlagi prvega obdelanega bloka izračuna potreben čas za izvedbo ene operacije za vsakega od odjemalcev. Na podlagi te ocene naj strežnik dodeljuje različnim odjemalcem tako dolge bloke, da jih bodo obdelali v približno enakem časovnem intervalu.

Naloga 13a
(10%)

Sestavite program, ki bo opravljal osnovno nalogo školjke (shell). Pogram bo analiziral ukazno vrstico, ki jo vtipka uporabnik, in izvršil ukaz. Po izvršitvi programa naj se z ukazom:

msh <enter>

na zaslonu izpiše znak pripravljenosti školjke na sprejem ukazne vrstice s tipkovnice terminala. Na primer:

msh[1]>

kjer je 1 zaporedna številka ukazne vrstice in se pri vsaki naslednji ukazni vrstici poveča za ena. Oblika ukazne vrstice naj bo:

msh[1]> ukaz [arg1] [arg2] …

kjer je *ukaz* ime ukaza in *arg1, arg2, …* argumenti tega ukaza.
Za izvršitev ukaza *ukaz* naj proces *msh* ustvari novi proces (s sistemskim klicem *fork*), ki naj se inicializira (z eno izmed izvedb sistemskega klica *exec*) iz datoteke z imenom *ukaz*. Proces *msh* naj med tem čaka, da se novoustvarjeni proces konča, nakar naj proces *msh* ponovno izpiše znak pripravljenosti *msh[i+1]>*.
Proces msh naj se zaključi, ko uporabnik vtipka ukaz *exit*.

Naloga 13b
(15%)

Sestavite program, ki bo opravljal nalogo školjke, tako kot program iz naloge 13a, z naslednjimi dodatki:

- Če je zadnji argument ukazne vrstice enak &, naj program *msh* ne čaka, da se novoustvarjeni proces konča, temveč naj takoj izpiše znak pripravljenosti *msh[i+1]>*.
- Program naj odpre datoteko z imenom *mhistory*, v katero naj spravlja vse ukaze, ki jih je uporabnik vtipkal. Če se uporabnikov ukaz glasi *hist*, naj izpiše zadnjih 10 ukazov, ki so shranjeni v datoteki *mhistory*.

Naloga 13c
(20%)

Sestavite program, ki bo opravljal nalogo školjke, tako kot program iz naloge 13b, z naslednjimi dodatki:

- Program naj omogoča preusmeritev standardnega izhoda v datoteko. Uporabnik preusmeri standardni izhod procesa, ki ga želi pognati tako, da kot zadnji argument vpiše >*ime*, pri čemer je > oznaka preusmeritve in *ime* ime izhodne datoteke.
- Program naj omogoča hkraten zagon dveh procesov, ki sta poveza na s cevjo. Ukazna vrstica naj se v tem primeru glasi:

  msh[1]> proc1 arg11 arg12 … | *proc2 arg21 arg22 …*

  pri čemer je *proc1* ime prvega ukaza, *arg11* in *arg12* argumenta prvega ukaza, | oznaka povezovanja (cev), *proc2* ime drugega ukaza ter *arg21* in *arg22* argumenta drugega ukaza. Standardni izhod procesa, ki ustreza prvemu ukazu, naj bo povezan s standardnim vhodom procesa, ki ustreza drugemu ukazu.

Naloga 14a
(10%)

Napišite program, ki izvršuje ukaze, ki so zapisani v tekstovni datoteki z imenom *command*. Datoteka vsebuje ukaze v obliki:

ukaz1 [arg11] [arg12] …<CR>
ukaz2 [arg21] [arg22] …<CR>
…

kjer je *ukaz1* ime prvega ukaza, *arg11, arg12, …* argumenti prvaga ukaza, *ukaz2* ime drugega ukaza, *arg21, arg22, …* argumenti drugega ukaza, *<CR>* oznaka za prehod v novo vrstico.
Za izvršitev posameznega ukaza naj se ustvari nov proces (s sistemskim klicem *fork*), ki naj se inicializira (z eno izmed izvedb sistemskega klica *exec*) iz datoteke z imenom ukaza. Izvrševanje naslednjega ukaza se lahko prične šele, ko se je prejšnji ukaz v celoti izvršil. Pred izvršitvijo ukaza naj se izpiše na terminal tekst *Začetek ukaza*, zaporedna številka ukaza in ukaz. Po izvršitvi ukaza naj se na terminalu izpiše tekst *Konec ukaza*, zaporedna številka ukaza ter rezultat izvršitve ukaza. Če se je proces končal zaradi sprejetega signala, naj bo rezultat izvršitve ukaza podan v obliki *signal : n*, pri čemer je *n* številka signala. Če se je proces normalno končal, naj bo rezultat izvršitve ukaza podan v obliki *rezultat : n*, pri čemer je *n* število, ki ga vrne izvršeni program pri izstopu.
Izvrševanje ukazov (in s tem tudi program) se konča, ko v datoteki ni več neizvršenih ukazov.

Naloga 14b
(15%)

Napišite program, ki izvršuje ukaze zapisane v datoteki, tako kot program iz naloge 14a, z naslednjo spremembo:
Program naj se po izvršitvi ukazov, ki jih vsebuje datoteka *command*, ne konča, ampak naj periodično (npr. vsako sekundo) preverja, ali je v datoteko vpisan novi ukaz. Če je, naj se ta ukaz izvrši. Postopek naj se nadaljuje, dokler se v datoteki ne pojavi ukaz exit. Upoštevajte, da je dovoljeno le dodajanje (append) novih ukazov v datoteko, tako da dolžina datoteke vedno narašča. Brisanje ali spreminjanje že vpisanih ukazov ni dovoljeno.