(SPaASM) Systémové programovanie - zadanie 2

Téma:

Systémové programovanie v C, systémové volania (Intel, Linux, FreeBSD), medziprocesová komunikácia.

Platforma:

Intel 386/Linux/C, alebo Intel 386/FreeBSD/C

Termín odovzdania:

9. cvičenie.

Hodnotenie:

15 bodov + bonus. V zmysle podmienok získania zápočtu minimálne **6** bodov. Výsledné hodnotenie je možné (okrem povinných častí) získať kombináciou rôznych úloh podľa vlastného výberu. To znamená, že okrem minima 6 bodov za povinné časti je možné ďalšie body (do 15, prípadne viac) získať za voliteľné časti ("voliteľné" a "nepovinné" neznamená to isté). Za voliteľné časti sa pripočíta maximálne 20 bodov. Maximálny počet všetkých možných získaných bodov za vyriešenie zadania je spolu 26 bodov.

Text zadania:

Napíšte v jazyku C jednoduchý interaktívny program, "shell", ktorý bude opakovane čakať na zadanie príkazu a potom ho spracuje. Na základe princípov klient-server architektúry tak musí s pomocou argumentov umožňovať funkciu servera aj klienta. Program musí umožňovať spúšťať zadané príkazy a bude tiež interpretovať aspoň nasledujúce špeciálne znaky: `# ; < > | \` . Príkazy musí byť možné zadať zo štandardného vstupu a tiež zo spojení reprezentovaných soketmi. Na príkazovom riadku musí byť možné špecifikovať prepínačom `-p` *port* číslo portu a/alebo prepínačom `-u` *cesta* názov lokálneho soketu na ktorých bude program čakať na prichádzajúce spojenia. Po spustení s prepínačom `-h` sa musia vypísať informácie o autorovi, účele a použití programu, zoznam príkazov. "Shell" musí poskytovať aspoň nasledujúce interné príkazy: *help* - výpis informácií ako pri `-h`, *quit* - ukončenie spojenia z ktorého príkaz prišiel, *halt* - ukončenie celého programu.

Prednastavený prompt musí pozostávať z mena používateľa, názvu stroja, aktuálneho času a zvoleného ukončovacieho znaku, e.g. '`16:34 user17@student#`'. Na zistenie týchto informácií použite vhodné systémové volania s použitím knižničných funkcií. Na formátovanie výstupu, zistenie mena používateľa z UID a pod. môžte v programe využiť bežné knižničné funkcie. Spúšťanie príkazov a presmerovanie súborov musia byť implementované pomocou príslušných systémových volaní. Tie nemusia byť urobené priamo (cez assembler), avšak knižničná funkcia `popen()`, prípadne podobná, nesmie byť použitá. Pri spustení programu bez argumentov, alebo s argumentom "-s" sa program bude správať vyššie uvedeným spôsobom, teda ako server. S prepínačom "-c" sa bude správať ako klient, teda program nadviaže spojenie so serverom cez socket, do ktorého bude posielať svoj štandardný vstup a čítať dáta pre výstup. Chybové stavy ošetrite bežným spôsobom. Počas vytvárania programu (najmä kompilácie) sa nesmú zobrazovať žiadne varovania a to ani pri zadanom prepínači prekladača `-Wall`.
Vo voliteľných častiach zadania sa očakáva, že tie úlohy budú mať vaše vlastné riešenia, nie jednoduché volania OS.

Odporúčané štúdium a povinné časti zadania:

Nasledujúce časti predstavujú **povinné minimum** pre akceptovanie funkčného zadania a po splnení budú hodnotené 6 bodmi.

- spracovanie argumentov, spracovanie zadaného vstupného riadku, interné príkazy *help*, *halt*, *quit*.
- overenie činnosti a spustenie zadaných príkazov, presmerovanie (volania `fork`, `exec`, `wait`, `pipe`, `dup`).
- sokety, spojenia (volania `socket`, `listen`, `accept`, `bind`, `connect`, `select`, `read`, `write`); systémové volania pre prompt.

Ďalšie úlohy

- voliteľné (možné získanie až 9 ďalších bodov výberom z voliteľných častí zadania až do 15 bodov)
- ostatné (možné získanie bonusových bodov výberom ďalších úloh z voliteľných častí zadania – max. 11 bonusových bodov)

Maximálny počet všetkých možných získaných bodov za vyriešenie zadania je teda spolu 26 bodov.

Voliteľné časti zadania:

Z nasledujúcich úloh môže študent vypracovať ľubovolný počet. Za vypracované úlohy sa pripočíta maximálne 20 bodov.

1. (2 body) Neinteraktívny režim - "shell" bude spracovávať aj príkazy v zadaných súboroch (skript).
2. (3 body) Program bude fungovať aj pod OS Linux aj pod FreeBSD (respektíve pod iným OS).
3. (3 body) Interný príkaz *stat* vypíše zoznam všetkých aktuálnych spojení na ktorých prijíma príkazy, prípadne aj všetky sokety na ktorých prijíma nové spojenia.
4. (2 body) Interný príkaz *abort n* ukončí zadané spojenie.
5. (4 body) Interné príkazy *listen* a *close* (s príslušnými argumentami) pre otvorenie a zatvorenie soketu pre prijímanie spojení.
6. (5 bodov) Podpora pre špeciálne znaky `` ` `` , nahradia sa výstupom príkazu ktorý obsahujú. Môže byť použitá funkcia `popen()`.
7. (3 body) Na zistenie informácií do prednastaveného promptu (meno užívateľa, názvu stroja, aktuálneho času a zvoleného ukončovacieho znaku) použte vhodné systémové volania priamo (napr. cez "inline assembler"), bez použitia knižničných funkcií.
8. (4 body) Presmerovanie výstupu do ľubovolného zvoleného deskriptoru, `>&`*n*, kde *n* je deskriptor súboru.
9. (3 body) S prepínačom "-c" v kombinácii s "-i", resp. "-u" sa bude program správať ako *klient*, teda pripojí sa na daný soket a bude do neho posielať svoj štandardný vstup a zobrazovať prichádzajúci obsah na výstup.
10. (2 body) Podpora pre špeciálne znaky `''`, zrušenie významu špeciálnych znakov medzi nimi (kvótovanie).
11. (2 body) S prepínačom "-i" bude možné zadať aj IP adresu na ktorej bude program očakávať spojenia (nielen port).
12. (3 body) Prepínače "-i", "-p" a "-u" bude možné zadať aj opakovane (viacnásobne pri jednom spustení), teda spojenia sa budú napríklad prijímať na viacerých portoch, alebo viacerých lokálnych soketoch.
13. (6 bodov) Vstup a výstup spúšťaných príkazov bude možné presmerovať aj do TCP spojení. Napríklad `ls >@ 127.0.0.1:1234`; `wc -l <@ 127.0.0.1:1234` .
14. (3 body) Konfigurovateľný tvar promptu, interný príkaz *prompt*.
15. (2 body) Podpora pre špeciálny znak `&`, nebude sa čakať na ukončenie spusteného príkazu.

16. (4 body) Podpora pre špeciálny znak `*`, nahradenie ľubovoľného podreťazca v názve súboru, nahradenie argumentu výsledkami vyhľadávania.
17. (2 body) Jeden z príkazov bude využívať funkcie implementované v samostatnej knižnici, ktorá bude "prilinkovaná" k hlavnému programu.
18. (5 bodov) Ak je niektoré spojenie nečinné zadanú dobu, bude zrušené.
19. (1 bod) Doba nečinnosti z predchádzajúceho bodu môže byť zadaná za argumentom "-t" a/alebo ako premenná prostredia.
20. (1 bod) S prepínačom "-v" sa budú zobrazovať pomocné (debugg-ovacie) výpisy na štandardný chybový výstup (stderr).
21. (2 body) Príkazy musia byť rozoznané aj ako argumenty na príkazovom riadku v kombinácii s prepínačom "-c" (interné príkazy ako prepínače, -halt, -help), vykonajú sa jednorazovo a program sa ukončí.
22. (2 body) Zmysluplné použitie premennej prostredia (e.g. prompt, log súbor, ...).
23. (4 body) Program s prepínačom "-d" sa bude správať ako démon (neobsadí terminál), nebude používať štandardný vstup a výstup.
24. (2 body) Program s prepínačom "-l" a menom súboru bude do neho zapisovať záznamy o vykonávaní príkazov (log-y).
25. (2 body) Program s prepínačom "-C" a menom súboru načíta konfiguráciu zo súboru (prompt, doba nečinnosti, log súbor, ...).
26. (1 bod) Predvolené meno konfiguračného súboru nastavené v premennej prostredia.
27. (5 bodov) Použitie signálov. E. g. znovunačítanie konfiguračného súboru po príchode zvoleného signálu, zachytenie Ctrl+C, vykonanie príkazu halt, quit a help (alebo iné).
28. (2 body) Funkčný Makefile.
29. (2 body) Vytvorenie a použitie konfiguračného skriptu (./configure) pre vytváranie programu.
30. (1 bod) Dobré komentáre, resp. rozšírená dokumentácia, v anglickom jazyku.

Poznámka:

Niektoré časti zadania môžu byť všeobecné, nejednoznačné a majú viacero správnych / vhodných riešení. Niektoré úlohy sú formulované len pre určitú platformu, ktorá bude zvolená na riešenie.

Zdroje informácií:

- Kompilácia a linkovanie: *man gcc*, riadenie kompilácie: *man make*, debug: *man gdb*,
- spôsob systémového volania: *man syscall*,
- systém *make* a jeho použitie: GNU Make,
- použitie assembleru v programe: GCC Inline Assembly Howto, Extended Asm - Assembler Instructions with C Expression Operands,
- práca v Linux-e, http://beej.us/guide/bgnet/html/ https://www.sallyx.org/sally/c/linux/
- práca v Unix-e, https://www.thegeekstuff.com/2009/09/how-to-write-compile-and-execute-c-program-on-unix-os-with-hello-world-example/
- FreeBSD Hypertext Man Pages - relevantná verzia je *5.2.1-RELEASE* (student), https://docs.freebsd.org/en/books/handbook/ https://docs.freebsd.org/doc/5.2.1-RELEASE/usr/share/doc/handbook/
- príklad vytvorenia shell-u: Tutorial - Write a Shell in C,
- doplnenie vedomostí z prednášok: Konvencia systémového volania - FreeBSD, Intel x86 Function-call Conventions - Assembly View, Call stack - Wikipedia.

Poznámka:

Na vypracovanie zadania je možné využiť aj prístup na server student.fiit.stuba.sk.