AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

Zdalne wykonywanie poleceń powłoki shell

z przedmiotu

Programowanie Sieciowe
Elektronika i Telekomunikacja, 3EiT

Michał Woźniak Jakub Pisarczyk

czwartek 17:50

prowadzący: Janusz Gozdecki

25.01.2022

1. Opis projektu

Implementacja usługi typu klient-serwer, do zdalnego wykonywania dwu członowych poleceń powłoki shell w trybie interaktywnym. Po stronie serwera, zgodnie z poleceniem zostały użyte funkcje takie jak pipe(), exec(), fork(), a wyniki przekazywane są do procesu serwera. Klient ma możliwość wywoływania różnych poleceń powłoki na innej maszynie poprzez sieć komputerową korzystając z protokołu TCP na porcie 514 (RSH). Po nawiązaniu połączenia przez rodzinę adresów IP wersji 6, klient poproszony jest o wpisanie 2 procesów, oddzielonych spacją, które chce wykonać potokowo, na przykład „date | cat" wykonamy poprzez wpisanie „date cat" oraz zatwierdzenie klawiszem ENTER. Wyświetli się informacja o wysłaniu polecenia do serwera, a klient będzie oczekiwał na kolejne polecenie do wysłania do czasu zakończenia działania. Serwer jawnie ignoruje sygnał SIG_CHLD. Po uruchomieniu oczekuje na połączenie, gdy klient wyśle polecenie, read() odczytuje dane z połączonego gniazda deskryptora, usuwa znak końca linii, otrzymany ciąg znaków przetwarzany jest na 2 osobne ciągi znaków, każdy z nich reprezentuje osobny proces, który będzie musiał się wykonać. Następne te dwa argumenty, po wywołaniu pipe() oraz fork(), przekazywane są do funkcji execlp(), serwer wyświetla wyniki wykonania odebranego polecenia, i jest gotowy na otrzymywanie kolejnych poleceń

2. Obrazowanie działania

Po połączeniu się klienta z serwerem wyświetla się powitalna grafika 'PS', wraz z krótkim opisem działania, oraz przykładowym wywołaniem programu

```
root@debian-ps:~/PROJEKT_KLIENT# ./klient fc00:1:1::1


Program pozwala wykonywać zdalne polecenia powłoki
       Shell na Serwerze z komputera klienta

               ^^^^        ^^^
               I   I      I
               I ^^        ^^
               I             I
               I          ^^^


       Michał Woźniak  /  Jakub Pisarczyk

Wpisz dwa procesy oddzielone spacją:
Np odpowiednio (ls | wc) -> (ls wc)
```

Parę przykładowych wywołań

```
Wpisz dwa procesy oddzielone spacją:
Np odpowiednio (ls | wc) -> (ls wc)
ls cat
Wysłano !
Wpisz kolejną komende powłoki Shell:
ls wc
Wysłano !
Wpisz kolejną komende powłoki Shell:
date wc
Wysłano !
Wpisz kolejną komende powłoki Shell:
date cat
Wysłano !
Wpisz kolejną komende powłoki Shell:
as df
Wysłano !
Wpisz kolejną komende powłoki Shell:
```

```
root@debian-ps:~/PROJEKT_SERWER2# ./test
------------------------------------ Nowe wywołanie ------------------------------------
a.out
test
test2
test2.c
test2.c.save
test.c
------------------------------------ Nowe wywołanie ------------------------------------
      6       6      45
------------------------------------ Nowe wywołanie ------------------------------------
      1       7      32
------------------------------------ Nowe wywołanie ------------------------------------
Thu 16 Dec 2021 09:50:39 PM CET
------------------------------------ Nowe wywołanie ------------------------------------
Filesystem                         1K-blocks    Used Available Use% Mounted on
udev                                  489560       0    489560   0% /dev
tmpfs                                 101108   11864     89244  12% /run
/dev/mapper/debian--ps--vg-root      6880992 2069460   4442280  32% /
tmpfs                                 505528       0    505528   0% /dev/shm
tmpfs                                   5120       0      5120   0% /run/lock
tmpfs                                 505528       0    505528   0% /sys/fs/cgroup
/dev/sda1                             240972   85355    143176  38% /boot
tmpfs                                 101104       0    101104   0% /run/user/0
```

KLIENT ***SERWER***

3. Podgląd na najważniejszą funkcję w kodzie serwera

```
void
zdalne_polecenia(int sockfd)
{
        ssize_t         n;
        char            buf[MAXLINE];    // Tablica charów o długości MAXLINE=1024
        //int           pozycja = 0;
        int             pipefd[2], status, done=0;
        pid_t           childpid;

again:
        while ( (n = read(sockfd, buf, MAXLINE)) > 0) {       // Czyta z gniazda sockfd dane, zapisuje w buf, n przyjmuje liczbe odczytanych bajtów

                buf[strlen(buf)-1] = '\0';     // Usuwanie znaku końca linii z bufora odczytanego z deskryptora

//////////////////////////////////////////////////////    SKRYPT

//    Skrypt zamieniający ciąg znaków zawierający spację w 2 ciągi znaków które stają się 2 parametrami

                char seps[] = " ,\t\n";    // Jeśli wystąpi któryś z podanych wyrażeń to przerywa ciąg znaków
                char *pch;
                pch = strtok(buf,seps);    // Zapisuje kawałek ciągu do tokena
                char *parametr_1 = pch;    // Patametr_1 otrzymuje wartość tego tokena - 1 człon komendy
                pch = strtok(NULL,seps);
                char *parametr_2 = pch;    // Patametr_2 otrzymuje kolejną wartość - 2 człon komendy
                pch = strtok(NULL,seps);
                char *parametr_3 = pch;

                char *duzo = "Za duzo parametrów! ";
                if(parametr_3 == ""){
                        fputs(duzo, stdout);
                }

                char *poczatek = "------------------------------------ Nowe wywołanie -------------------------------------\n";
                fputs(poczatek, stdout);

//////////////////////////////////////////////////////        OBSŁUŻENIE PIPE() FORK() EXEC()

                pipe(pipefd);

                childpid = fork();    // fork() - w celu zapewnienia komunikacji między procesem macierzystym oraz jego procesami potomnymi
                if (childpid == 0) {
                        dup2(pipefd[1],STDOUT_FILENO);
                        execlp(parametr_1, parametr_1, (char *) NULL);
                }
                childpid = fork();
                if (childpid == 0) {
                        close(pipefd[1]);
                        dup2(pipefd[0],STDIN_FILENO);
                        execlp(parametr_2, parametr_2, (char *) NULL);
                 }

                close(pipefd[0]);   // deskryptor pliku tylko do odczytu
                close(pipefd[1]);   // deskryptor pliku tylko do zapisu

////////////////////////////////////////////////////////////////////////////////////////
        }
        if (n < 0 && errno == EINTR)     // Obsługa przerwania, wznawia funkcje
                goto again;
        else if (n < 0)
                perror("zdalne_polecenia: BŁĄD ODCZYTU");
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```