Instrukcja do zajęć laboratoryjnych

Język ANSI C (w systemie LINUX)

wersja: 1.31

Nr ćwiczenia:	1	
Temat:	**Podstawy pracy w systemie LINUX**	
Cel ćwiczenia:	Celem ćwiczenia jest zapoznanie studenta z organizacją laboratorium oraz podstawami pracy w środowisku LINUX w kontekście nauki programowania w języku ANSI C.	
Wymagane przygotowanie teoretyczne:	Informacje podane na wykładzie.	
Sposób zaliczenia:	Sprawozdanie w formie pisemnej.	[ ]
	Pozytywna ocena ćwiczenia przez prowadzącego pod koniec zajęć.	[x]

1. Konwencje przyjęte w instrukcji

`Czcionka o stałej szerokości`

Nazwy programów, poleceń, katalogów, wyniki działania wydawanych poleceń.

`Czcionka o stałej szerokości pogrubiona`

Podaje tekst, który należy dosłownie przepisać. W przypadku plików źródłowych wyróżnia istotniejsze fragmenty.

`Czcionka o stałej szerokości kursywą`

Tekst komentarza w przykładowych sesjach przy terminalu.

`Czcionka o stałej szerokości kursywą pogrubiona`

Wyróżnia istotniejsze fragmenty wyników działania wydawanych poleceń.

2. Uwagi wstępne

Celem pierwszego ćwiczenia jest zapoznanie studenta z najważniejszymi zasadami pracy w systemie LINUX. Ponieważ systemy operacyjne z rodziny UNIX będą (były lub są) tematem innego przedmiotu, dlatego też podane zostanie tylko minimum informacji niezbędnych do

rozpoczęcia programowania w języku C w środowisku systemu LINUX. Student powinien samodzielnie, stopniowo zapoznawać się z tym systemem operacyjnym. Poniżej podano tylko najważniejsze informacje, które należy traktować jako wstęp do bardziej wnikliwego poznawania systemu LINUX.

Zakładamy, że praca z systemem będzie odbywać się **wyłącznie** w „czystej" postaci, czyli bez pośrednictwa żadnego menadżera okien (jak na przykład GNOME, KDE, czy też inne). Wszystkie polecenia będą więc wydawane z konsoli a używane programy będą pracować w trybie tekstowym (choć niektóre z nich, jak np. bardzo popularny `Midnight Commander`, będą posiadały namiastkę grafiki, tzw. semigrafikę).

3. Rozpoczęcie pracy w środowisku LINUX. Logowanie i kończenie pracy

Po rozpoczęciu zajęć prowadzący poda każdemu studentowi:

- nazwę serwera,
- nazwę konta,
- hasło dostępowe,

do systemu LINUX. Każdy student otrzyma indywidualne konto, które nie będzie zmieniane w czasie trwania kursu. Nie będzie też kasowana zawartość jego katalogu roboczego.

Jakkolwiek podłączenie się do serwera LINUX możliwe jest na wiele różnych sposobów i z użyciem wielu różnych programów, sugerujemy aby używać do tego celu programu o nazwie `PuTTY` (jest to bezpłatny program, który można pobrać ze strony WWW: `http://www.chiark.greenend.org.uk/~sgtatham/putty/`). Jego niewątpliwą zaletą jest bardzo prosta, wręcz intuicyjna obsługa. Po prawidłowym zalogowaniu się powinieneś zobaczyć ekran podobny do pokazanego poniżej:

```
login as: lab1
lab1@mykonos.iie.uz.zgora.pl's password:
Linux mykonos 2.4.18-bf2.4 #1 Son Apr 14 09:53:28 CEST 2002 i686 unknown

*****************************************
*           Witamy na serwerze          *
*         mykonos.iie.uz.zgora.pl       *
*             Debian GNU/Linux          *
*                                       *
*         Uniwersytet Zielonogorski     *
*     Instytut Informatyki i Elektroniki*
*****************************************

Czytam plik:  /etc/profile
lab1@mykonos:~$
```

Wygląd ekranu (czyli np. użyta czcionka, kolorystyka itd.) można z łatwością zmienić konfigurując odpowiednio program `PuTTY`. Ponieważ obsługa tego programu jest bardzo intuicyjna proponujemy samodzielnie dostosować wygląd ekranu do swoich upodobań.

Katalogiem domowym każdego użytkownika jest odpowiedni podkatalog w katalogu `/home` (np. dla użytkownika `lab1` będzie to `/home/lab1`). „Na dzień dobry" w każdym z tych katalogów znajdują się pliki ukryte (o nazwach zaczynających się od znaku kropki) `.bash_profile` oraz `.bashrc` (ten drugi jest wywoływany z wnętrza tego pierwszego). Wszystkie wpisane tam polecenia będą wykonywane automatycznie w czasie logowania się użytkownika do systemu.

Można wiec wpisywać tam swoje własne polecenia dostosowujące środowisko pracy do indywidualnych potrzeb.

Aby zakończyć pracę z systemem należy wydać polecenie `exit` lub `logout` (czasami można również użyć kombinacji klawiszy `Ctrl-D`).

Automatyczne uzupełnianie nazw

Pracując z systemem LINUX bardzo często musimy wpisywać dość długie nazwy plików i komend (zwłaszcza te pierwsze potrafią być naprawdę baaaaaardzo długie). Aby nie „wklepywać" ich za każdym razem możemy wykorzystać bardzo prosty i jednocześnie bardzo funkcjonalny mechanizm *automatycznego uzupełniania nazw*.

Po wpisaniu początkowego fragmentu nazwy wciskamy klawisz `Tab`. Jeżeli ten początkowy fragment jednoznacznie wskazuje na dokładnie jeden plik lub dokładnie jedną komendą następuje automatyczne uzupełnienie nazwy. Gdy natomiast początkowy fragment może zostać skojarzony z dwoma lub więcej plikami lub komendami to system, po powtórnym naciśnięciu klawisza `Tab`, wyświetli wszystkie nazwy „pasujące" do podanego wzorca. Możemy wówczas doprecyzować zapytanie wpisując kolejne znaki nazwy i ponownie nacisnąć klawisz `Tab`.

Jeżeli wprowadzany tekst jest pierwszym słowem w linii to system zakłada, że jest to nazwa komendy i próbuje wówczas dopasować do podanego wzorca te komendy, które są dostępne w lokalizacjach określonych przez zmienną `PATH`.

Poniżej pokazano fragment sesji przy terminalu. Tab oraz Enter oznacza naciśnięcie klawisza `Tab` oraz `Enter`.

```
$ ls Enter
moj_plik1  moj_plik2  plik1  plik2

Ponieważ fraza 'pl' jest pierwszym słowem w linii, więc LINUX traktuje je
jako nazwę komendy. Wyświetlane są więc wszystkie komendy, które
rozpoczynają się od frazy 'pl'.

$ pl Tab Tab
pl2pm  pldes  plog
$ pl

Teraz (przy pomocy komendy cat) chcemy wyświetlić zawartość wybranego
pliku. Po wpisaniu frazy 'p' i naciśnięciu Esc LINUX automatycznie
uzupełni nazwę do 'plik'. Ponieważ 'moj_plik1' oraz ' moj_plik2' nie
pasują do wzorca są pomijane przy wyświetlaniu.

$ cat p Esc
$ cat plik Tab Tab
plik1  plik2

Dopisujemy do nazwy '1' i wyświetlamy zawartość pliku.

$ cat plik1 Enter
Oto jest zawartosc pliku 'plik1'
$
```

4. Najważniejsze polecenia systemu operacyjnego

Poniżej zebrano kilkanaście najważniejszych poleceń systemu LINUX, bez znajomości których praktycznie nie sposób pracować z systemem. Każde z tych poleceń posiada pewną liczbę opcji (czasem dość pokaźną!), których szczegółowy opis można znaleźć w dokumentacji systemowej `man` – patrz opis w kolejnym rozdziale. Opcje, często jednoliterowe, umieszczane są zawsze za nazwą polecenia i poprzedzone są znakiem minusa. Należy samodzielnie przećwiczyć każde z tych poleceń[1].

```
ls, cp, rm, rmdir, mv, mkdir, touch, find, cat, grep, sort, cd, chmod,
pwd, head, tail, id, info, man, more, less, passwd, logout, exit
```

Uwaga: w systemie LINUX (i każdym innym z rodziny UNIX) rozróżniane są małe i duże litery, więc polecenie `ls` to nie to samo co `LS` lub `Ls` (te dwa ostatnie nie występują po prostu w standardowej instalacji systemu – choć oczywiście nic nie stoi na przeszkodzie, aby napisać samodzielnie programy o takich właśnie nazwach).

Poniżej podano polecenia, które należy wykonać. Należy do każdego z nich „dopasować" odpowiednie polecenie z podanych powyżej. Jeżeli uda Ci się **świadomie** wszystkie je wykonać możesz przyjąć, że masz opanowany podstawowy kanon poleceń systemu UNIX (LINUX).

- zmienić swoje hasło dostępowe,
- wyświetlić dane o sobie (nazwę konta, grupę lub grupy, do których należy zalogowany użytkownik),
- wyświetlić zawartość swojego katalogu roboczego. Użyć opcji `-a`, `-l`, `-C`, `-F`, `-R` odpowiedniego polecenia i zaobserwować ich działanie,
- utworzyć w swoim katalogu domowym (czyli `/home/nazwa_konta`) katalog o nazwie `kat1` a w nim jeszcze jeden katalog o nazwie `podkat1`,
- wyświetlić informację o katalogu, gdzie w danej chwili jesteśmy (czyli katalogu bieżącym),
- przejść do katalogu `podkat1`. Następnie przejść do katalogu głównego (ang. *root directory*, oznaczanego znakiem ukośnika `/`) i z powrotem do katalogu domowego (szybki sposób przejścia do katalogu domowego to użycie znaku tyldy `~`). Katalog domowy przechowywany jest w zmiennej środowiskowej `$HOME`,
- utworzyć w katalogu `podkat1` plik `pierwszy.txt` i wpisać do niego dowolny tekst. Zamknąć ten plik i ponownie wczytać go do dowolnego edytora tekstowego. Dopisać do niego dowolny tekst (tak, aby było w sumie około 50 linii tekstu),
- wyświetlić 5 pierwszych linii tekstu z pliku `pierwszy.txt`,
- wyświetlić 5 ostatnich linii tekstu z pliku `pierwszy.txt`,
- wyświetlić na ekranie te linie z pliku `pierwszy.txt`, które zawierają słowo „LINUX" (zakładamy oczywiście, że w pliku to słowo się znajduje. W przeciwnym wypadku polecenie oczywiście nic nie wyświetli na ekranie),
- zmienić nazwę pliku `pierwszy.txt` na `drugi.txt`,
- skopiować plik `drugi.txt` do katalogu `kat1`

[1] Niektóre z podanych tu poleceń są bardzo proste, inne dosyć rozbudowane i o bardzo wielu możliwościach (opcjach).

- dla pliku `$HOME/kat1/drugi.txt` odebrać prawa dostępu dla wszystkich innych użytkowników niż właściciel pliku. Jak przekonać się, że prawa zostały rzeczywiście odebrane?
- usunąć katalogi `kat1` oraz `podkat1` wraz z zawartością,
- nadać dla wszystkich użytkowników prawo do czytania dla dowolnego pliku w swoim katalogu domowym. W odpowiednim poleceniu użyć zarówno trybu symbolicznego jak i ósemkowego do określania praw dostępu,
- znaleźć na dysku wszystkie pliki, których nazwa rozpoczyna się frazą „lin". Znalezioną listę zapisać do pliku z pominięciem ewentualnych komunikatów o błędach,
- odnaleźć na dysku wszystkie pliki, których rozmiar mieści się w zakresie od 100kb do 500kb i nazwy tych plików wyświetlić poleceniem `ls`. Polecenie napisać w taki sposób, aby wszelkie ew. komunikaty o błędach (np. związane z brakiem uprawnień do przeglądania plików) były ignorowane, tzn. nie pojawiały się na ekranie. (wskazówka: użyć przełączników `-size`, `-exec` oraz przekierowania `2>/dev/null`),
- posortuj alfabetycznie (!?) treść manual-a (polecenie `man`) polecenia `passwd`,
- prawidłowo zakończyć pracę w systemie LINUX.

Przekierowania i potoki

Większość poleceń zawartych w ramce powyżej może korzystać z bardzo efektywnych operatorów systemowych jakim są tzw. *przekierowania i potoki*. Bez możliwości ich stosowania, większość poleceń systemu Linux byłaby bardzo uboga i mało użyteczna.

Każdy działający w systemie proces (program) ma przypisany do siebie jeden lub więcej tzw. strumieni danych. W systemie UNIX strumieniem określa się źródło danych, na których pracuje proces (program) lub miejsce, do którego proces wysyła dane będące efektem jego działania. Takim strumieniem może być dowolny plik, czyli na przykład plik dyskowy, ekran monitora, klawiatura[2]. Domyślnie dla każdego procesu tworzone są trzy tzw. *strumienie standardowe*:

- standardowy strumień wejściowy o numerze 0 (ang. *standard input*) oznaczany przez `stdin`,
- standardowy strumień wyjściowy o numerze 1 (ang. *standard output*) oznaczany przez `stdout`,
- standardowy strumień błędów o numerze 2 (ang. *standard error*) oznaczany przez `stderr`.

Podczas normalnej pracy procesy (programy) pobierają dane ze standardowego strumienia (domyślnie klawiatura), a wyniki wysyłają na standardowe wyjście (domyślnie na ekran). Przypisanie strumieni wejściowych i wyjściowych można jednak prosto zmienić (przekierować) przed uruchomieniem procesu. Służą do tego operatory `<`, `>`, `<<`, `>>`.

```
Wypisuje zawartość pliku na ekranie

$ cat plik.txt
```

[2] W systemie UNIX klawiatura i ekran traktowane są jak pliki, z których można czytać dane i do których można wysyłać dane.

Zmieniono standardowe wyjście z ekranu (który z punktu widzenia systemu operacyjnego jest plikiem) na „własny" plik o nazwie plik.txt. Standardowego wejścia nie zmieniliśmy, więc system oczekuje na dane ze standardowego pliku wejściowego, czyli z klawiatury.

$ **cat > plik.txt**
Piszemy cokolwiek
Ctrl–D

To co napisaliśmy trafia do wskazanego pliku, czyli do plik.txt. Plik plik.txt zostanie więc utworzony i po naciśnięciu Ctrl–D zapisany na dysku.
Uwaga: Ta wersja tworzenia pliku jest trochę niebezpieczna. Gdy ponownie wydamy polecenie i gdy plik.txt już istnieje, to zostanie on usunięty i utworzony na nowo – stracimy więc całą jego dotychczasową zawartość!

$ **cat > plik.txt**

Teraz zmieniamy standardowe wejście. Dane zamiast z klawiatury będą pobierane z pliku.

$ **cat < plik.txt**

Teraz zmieniamy standardowe wejście i wyjście jednocześnie. Dane z jednego pliku zostaną skierowane do drugiego pliku.

$ **cat < wej.txt > wyj.txt**

Teraz zamiast nadpisywać wyniki, dopiszemy je do pliku.

$ **cat >> plik.txt**
Piszemy cokolwiek
Ctrl–D

Gdy wyświetlamy zawartość np. 300 plików i w czasie wykonywania tej czynności 18. plik nie istnieje, to pojawi się komunikat o błędzie. W wielkiej ilości informacji wyświetlających się na ekranie prawdopodobnie tego nie zauważymy. Możemy jednak skorzystać ze standardowego strumienia błędów. Ewentualne komunikaty o błędach znajdą się w pliku err.txt.

$ **cat 1.txt 2.txt ... 456.txt 2> err.txt**
Uwaga: *kropki nie są elementem polecenia!*

W tej wersji ewentualne komunikaty zostaną, poprzez przesłanie ich do specjalnego strumienia o nazwie null, zignorowane.

$ **cat 1.txt 2.txt ... 456.txt 2> /dev/null**

W tej wersji wynik pojawi się w pliku out.txt a ewentualne komunikaty zostaną tak jak poprzednio zignorowane.

$ **cat 1.txt 2.txt ... 456.txt > out.txt 2> /dev/null**

Bez przekierowania dla zwykłego użytkownika na ekranie będzie pojawiać się bardzo dużo informacji o braku praw odczytu do katalogu. Aby „odsiać" te błędy, pozbędziemy się ich przez odpowiednie przekierowanie strumienia.

$ **find /usr –name abc* 2> /dev/null**

Strumienie procesów można ze sobą łączyć, tworząc tzw. *potoki*. Przykładowo dane wyjściowe z jednego procesu mogą stać się danymi wejściowymi drugiego procesu. Do obsługi potoków służy operator | (pionowa kreska).

```
Polecenie more zatrzymuje (strona po stronie) informacje pojawiające się
na ekranie (bardziej ogólnie: będące jego argumentem). Łatwo możemy więc
obejrzeć zawartość „dużego" katalogu.

$ ls -la /usr/bin | more

Wyświetlamy najpierw pierwsze 10 linii z pliku plik.txt a następnie z tych
10 linii wyświetlamy pięć ostatnich linii.

$ cat plik.txt | head -n 10 | tail -n 5

Błędy trafiają NIE do potoku, ale (domyślnie) na ekran. Na ekranie pojawi
się więc pierwsze 10 znalezionych plików oraz komunikaty o błędach.

$ find /usr -name abc* | head -n 10

Teraz „odsialiśmy" błędy.

$ find /usr -name abc* 2> /dev/null | head -n 10

Teraz komunikaty o błędach obsługujemy na równi z wynikami polecenia find.
Zarówno wyniki polecenia find jak i komunikaty o błędach przesłaliśmy do
strumienia jednego stdout (o numerze 1). Następnie dane w tym strumieniu
„obrabiamy" poleceniem head.

$ find /usr -name abc* 2>&1 | head -n 10

Istnieje możliwość łączenia całego szeregu potoków. Poniższe polecenie
wyświetli objętość w megabajtach (przełącznik -m) wszystkich podkatalogów
w naszym katalogu domowym użytkownika (polecenie du i symbol tyldy ~),
wynik zostanie posortowany w odwróconym porządku (przełącznik -r) i według
wartości liczbowych łańcuchów (przełącznik -n). Na ekranie pojawi się
pierwszych 20-tu „rekordzistów" jeśli chodzi o zajętość dysku  (polecenie
head)

$ du -b ~ | sort -rn | head -20

Teraz dodatkowo do pliku wyślemy wszystkie wyniki, a na ekran (tak jak
poprzednio) tylko pierwsze 20 pozycji. Skorzystano tu z bardzo użytecznego
narzędzia tee.

$ du -b ~ | sort -rn | tee plik.txt | head -20
```

5. Pomoc systemowa

Podstawowym źródłem wiedzy na temat systemu UNIX (LINUX) (oprócz oczywiście książek, których jest na rynku księgarskim bardzo dużo i każdy student powinien wybrać sobie taką / takie, które jemu najbardziej odpowiadają) jest system pomocy `man` (ang. *manual*). Jest to zestaw dokumentów tekstowych opisujących różne aspekty systemu. Jako parametr polecenia należy podać nazwę polecenia, dla którego pomoc ma być przywołana, np:

```
$ man ls
```

Ponieważ plików z pomocą systemową jest bardzo wiele, podzielono je na grupy (sekcje). Każda grupa zawiera opisy pewnego typu. Przykładowo opis formatów plików konfiguracyjnych znajduje

się w grupie 5. Chcąc przykładowo przywołać pomoc na temat formatu pliku `passwd` należy wydać następujące polecenie:

```
$ man 5 passwd
```

Instrukcja obsługi samego programu `man` dostępna jest również w ten sam sposób, czyli po wydaniu następującego polecenia:

```
$ man man
```

Uwaga: po odpowiednim skonfigurowaniu systemu LINUX możliwe jest wyświetlanie tekstów pomocy systemowej w różnych językach, w tym też w języku polskim. Wymaga to jednak odpowiedniego ustawienia zmiennej środowiskowej `LANG`. W przypadku języka polskiego oraz powłoki `bash` (jeżeli nie wiesz co to jest powłoka oraz zmienne środowiskowe spytaj się kogoś bardziej doświadczonego) należy wykonać komendę:

```
$ export LANG=pl_PL
```

Od tej pory większość tekstów dokumentacji będzie wyświetlała się w języku polskim. Powrót do oryginalnej wersji angielskiej uzyskamy wydając jedno z dwóch poniższych poleceń:

```
$ export LANG=eng_ENG
$ export LANG=c_C
```

Nieco inne podejście do dokumentacji systemowej prezentuje system `info`. Program `info` należy uruchomić, podobnie jak w przypadku programu `man`, z parametrem wskazującym na poszukiwane słowo kluczowe. Wywołanie programu bez parametru przywoła indeks zawartości systemu `info`. Proponujemy samodzielnie zapoznać się z tym poleceniem.

Na koniec należy również wspomnieć o tzw. plikach *HOWTO*. Jest to ogromna baza wiedzy na temat systemu LINUX. Opisane są tam w szczegółach różne kwestie dotyczące działania systemu. Wiele z tych dokumentów zostało przetłumaczonych na język polski. W dystrybucji o nazwie *Debian* znajdują się one w katalogu `/usr/share/doc/HOWTO`. Na początek warto zapoznać się z dwoma dokumentami napisanymi specjalnie dla tych użytkowników systemu LINUX, którzy do tej pory pracowali jedynie z systemami DOS / Windows. Są to pliki:

- `/usr/share/doc/HOWTO/pl-txt/dos2Linux-HOWTO.pl.txt.gz`
- `/usr/share/doc/HOWTO/en-txt/DOS-Win-to-Linux-HOWTO.gz`

6. Edytory

Tworząc i edytując pliki tekstowe musimy używać jakiegoś edytora tekstowego. W systemie LINUX istnieje pokaźna liczba tego typu programów. Każdy powinien wybrać sobie najbardziej mu odpowiadający (wybór jest jednak zwykle ograniczony do tych, które zainstalował administrator – zawsze jednak można spróbować poprosić go o zainstalowanie dodatkowego edytora). Proponujemy zapoznać się z następującymi edytorami:

- `vi` – klasyka gatunku. Istniał w systemie UNIX „od zawsze” i wstyd po prostu go nie znać. W wielu dystrybucjach LINUX-a edytor ten jest zastąpiony nieco unowocześnioną jego wersją o nazwie `vim`. Wersja ta zachowuje oryginalną koncepcję edytora `vi`, jednak wprowadza wiele udogodnień (jak np. możliwość podświetlania składni, wielopoziomowe *undo* i wiele innych – patrz system pomocy `man`),
- `ed` – obecnie, ze względu na archaiczny sposób obsługi, trąci nieco myszką. Znajomość jego przydaje się jednak wówczas, gdy mamy awarię systemu i żaden inny edytor nie jest w danym momencie dostępny. Aby jednak poczuć ducha historii, proponujemy choćby pobieżnie zapoznać się z nim,

- `joe` – całkiem wygodny i przyjazny w użyciu edytor,
- `elvis` – jak wyżej,
- `Midnight Commander` – to w zasadzie LINUX-owa wersja znanego pewnie wszystkim 'klasycznego" programu Norton Commander (bez niego świat informatyki wyglądałby inaczej, prawda `;-)`). Ma jednak wbudowany bardzo przyjazny edytor, który ma tą przyjemną cechę, że umie podświetlać w różnych kolorach słowa kluczowe w języku C (oraz innych języków programowania). Uruchamiamy go poleceniem `mc`.

7. Program FTP

Pracując na serwerze LINUX, bardzo często będziemy mieć potrzebę kopiowania plików z oraz do serwera (np. z dyskietki do swojego katalogu domowego lub odwrotnie). Najwygodniej użyć do tego klasycznego programu `ftp`. Student powinien samodzielnie zapoznać się z podstawowymi jego komendami (poniżej podano te najważniejsze). Więcej informacji można znaleźć na stronach pomocy systemowej `man ftp`.

```
ls, pwd, cd, lcd, put, get mput, mget, hash, binary, ascii, prompt, quit,
bye, close,
```

Przykładowa sesja na komputerze może wyglądać następująco:

```
Z maszyny Windows-owej uruchamiamy program ftp i łączymy się do serwera
LINUX-owego. Logujemy się na użytkownika lab1.

C:\>ftp mykonos.iie.uz.zgora.pl
Connected to mykonos.iie.uz.zgora.pl.
220 mykonos FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
User (mykonos.iie.uz.zgora.pl:(none)): lab1
331 Password required for lab1.
Password:
230- Linux mykonos 2.4.18-bf2.4 #1 Son Apr 14 09:53:28 CEST 2002 i686
unknown
230-
230- ********************************************
230- *            Witam na serwerze             *
230- *         mykonos.iie.uz.zgora.pl          *
230- *             Debian GNU/Linux             *
230- *                                          *
230- *         Uniwersytet Zielonogorski        *
230- *     Instytut Informatyki i Elektroniki   *
230- ********************************************
230-
230 User lab1 logged in.

Patrzymy jaki jest nasz bieżący katalog na zdalnej maszynie.

ftp> pwd
257 "/home/lab1" is current directory.

Wyświetlamy zawartość katalogu bieżącego na zdalnej maszynie.

ftp> ls -la
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 24
drwxr-xr-x    2 lab1     students     4096 Sep 22 10:14 .
drwxrwsr-x    4 root     staff        4096 Sep 22 10:14 ..
-rw-r--r--    1 lab1     students      266 Sep 22 10:14 .alias
```

```
-rw-r--r--    1 lab1     students      509 Sep 22 10:14 .bash_profile
-rw-r--r--    1 lab1     students     1093 Sep 22 10:14 .bashrc
-rw-r--r--    1 lab1     students      375 Sep 22 10:14 .cshrc
226 Transfer complete.
ftp: 393 bytes received in 0,00Seconds 393000,00Kbytes/sec.

Ustawiamy katalog bieżący na lokalnej maszynie.

ftp> lcd a:\
Local directory now A:\.

Ustawiamy tryb przesyłu na binarny.

ftp> binary
200 Type set to I.

Przesyłamy plik z katalogu bieżącego na maszynie lokalnej (dysk a:\) do
katalogu bieżącego na maszynie zdalnej (katalog /home/lab1).

ftp> put tematy.doc
200 PORT command successful.
150 Opening BINARY mode data connection for 'tematy.doc'.
226 Transfer complete.
ftp: 41472 bytes sent in 0,76Seconds 54,50Kbytes/sec.

Sprawdzamy, czy plik został przesłany na maszynę zdalną.

ftp> ls -la
total 68
drwxr-xr-x    2 lab1     students     4096 Sep 22 10:36 .
drwxrwsr-x    4 root     staff        4096 Sep 22 10:14 ..
-rw-r--r--    1 lab1     students      266 Sep 22 10:14 .alias
-rw-r--r--    1 lab1     students      509 Sep 22 10:14 .bash_profile
-rw-r--r--    1 lab1     students     1093 Sep 22 10:14 .bashrc
-rw-r--r--    1 lab1     students      375 Sep 22 10:14 .cshrc
-rw-r-----    1 lab1     students    41376 Sep 22 11:04 tematy.doc

Z katalogu bieżącego maszyny zdalnej pobieramy plik i przesyłamy go do
katalogu bieżącego na maszynie lokalnej.

ftp> get .bash_profile
200 PORT command successful.
150 Opening BINARY mode data connection for '.bash_profile' (509 bytes).
226 Transfer complete.
ftp: 509 bytes received in 0,42Seconds 1,21Kbytes/sec.
```

Literatura

Uwaga: literatura na temat postaw działania systemu LINUX jest bardzo obszerna. Nie sposób podać tu choćby jej części. Poniższa lista kilku pozycji powinna być potraktowana jako subiektywny wybór autora niniejszej instrukcji a nie jako zalecane czy też obowiązujące pozycje.

1. Cezary Sobaniec. *System operacyjny LINUX – przewodnik użytkownika.* Wydawnictwo NAKON, Poznań, 2002.

2. Wiesława Bartol, Małgorzata Lisowiec, Jan Ogłaza. *Unix kurs użytkownika.* Oficyna Wydawnicza READ ME, Warszawa, 1994.

3. P. Silvester. *System operacyjny UNIX*. PWN, Warszawa, 1990.

4. Z. Królikowski, M. Sajkowski. *System operacyjny UNIX dla początkujących i zaawansowanych*. Wydawnictwo NAKOM, Poznań, 1995.