

Podstawy systemu UNIX

© ⓘ Ⓢ Paweł Kędzierski & Robert Góra

3 kwietnia 2019

1 Prawa dostępu do plików

W systemach *NIX, czyli rodzinie systemów unixowych, wszystkie urządzenia fizyczne mają odpowiadające im pliki w systemie plików. Każdy plik ma też przypisany zestaw uprawnień i właściciela. Informacje te można wyświetlić komendą `ls -l`:

```
drwxr-xr-x 2 rgora kdm      2048 03-26 16:39 n2h2/
-rw-r--r-- 1 rgora kdm      4552 03-26 12:12 scan.dat
lrw-r--r-- 1 rgora kdm        75 03-26 12:14 scan.gpl
```

Poszczególne kolumny oznaczają (od lewej):

1. typ pliku i prawa dostępu;
2. liczba *dowiązań* do pliku;
3. właściciel pliku;
4. grupa, do której należy plik;
5. rozmiar w bajtach;
6. data ostatniej modyfikacji;
7. nazwa pliku.

Znaczenie kodu w pierwszej kolumnie jest następujące:

-	rwX	r-X	r--
typ pliku	prawa właściciela	prawa grupy	prawa innych
Typ pliku		Prawa dostępu	
d	katalog	r	prawo do czytania (read)
-	zwykły plik	w	prawo do zapisu (write)
l	skrót (link)	x	prawo do wykonywania (eXecute)

Prawo odczytu oznacza możliwość obejrzenia zawartości oraz utworzenia kopii pliku. Prawo zapisu do pliku oznacza możliwość jego modyfikacji, ale nie skasowania — aby skasować plik (lub utworzyć nowy), trzeba mieć prawo do zapisu w katalogu, w którym ten plik się znajduje. Wreszcie prawo do wykonywania oznacza możliwość uruchomienia pliku jako programu.

Nieco inne jest znaczenie praw dostępu w przypadku katalogów. Prawo do odczytu katalogu oznacza, że można sprawdzić, jakie pliki są w tym katalogu, ale nie można uzyskać o nich żadnej innej informacji — do tego trzeba mieć prawo wykonywania katalogu. Prawo wykonywania jest również niezbędne, aby móc przejść do katalogu (komendą `cd`) oraz aby mieć dostęp do plików tam zapisanych (niezależnie, a raczej nadrzędnie do ich własnych praw dostępu).

2 Podstawowe komendy w systemie Unix

Po zalogowaniu się w systemie Unix, lub przy otwarciu terminala tekstowego w sesji X-Windows, uruchomiony zostaje program odpowiedzialny za komunikację z użytkownikiem, tzw. `shell`. Jego zadaniem jest interpretacja komend wydawanych przez użytkownika i ich wykonanie.

Powłoka interpretuje komendy po wpisaniu linii (ewentualnych poprawkach) i wciśnięciu klawisza Enter. Linia dzielona jest następnie na wyrazy (wg odstępów); pierwszy wyraz w linii jest traktowany jako komenda, kolejne wyrazy jako opcje lub argumenty:

```
komenda -opcja -opcja argument1 argument2
```

Opcje tradycyjnie podawane są po znaku „-” (minus). Jednak to, jakie opcje i argumenty są rozumiane lub wymagane przez konkretną komendę, zależy już tylko i wyłącznie od niej samej. Najczęściej, opcja `-h` (w systemie Linux także `--help`) wyświetla krótki opis komendy. Pełna instrukcja jest dostępna przez komendę `man` (ang. manual), np.

```
man man
```

podaje informacje, jak działa sam manual. Informacje wypisywane przez `man` są przepuszczane przez program `more` lub `less` (tzw. pager), pozwalający na obejrzenie ich strona po stronie. `more` i `less` rozumieją klawisze Enter (kolejna linia), Spacja (kolejna strona) oraz Q (quit, koniec). `less` jest inteligentniejszy, rozumie także klawisze strzałek, PgUp, PgDn itd., i pozwala na przewijanie w przód i w tył. Obydwa programy można samodzielnie wykorzystać do oglądania wyników wypisywanych przez jakąś komendę, lub zawartości plików:

```
more jakis_plik
ls --help | less
```

Użyłem tutaj znaku „|”, czyli tzw. *rury* (ang. pipe) — znak ten jest bowiem przez shell rozumiany jako wymóg przesłania wyników komendy do innego programu. Specjalne znaczenie mają dla shell-a także znaki „<” i „>” (tzw. strzałki) — pozwalają one skierować strumień danych z programu do pliku (>) lub z pliku do programu (<).

```
ls -l > lista_plikow.txt
sort < lista_plikow.txt
```

Innym ważnym znakiem specjalnym jest „&” (ampersand). Dodany na końcu komendy powoduje, że shell uruchamia tę komendę w *tle*, czyli nie czekając na jej zakończenie powróci do stanu gotowości do wykonywania nowych komend. Tak należy uruchamiać programy, które otwierają własne okno X-Windows, aby nie blokowały niepotrzebnie terminala; w ten sam sposób uruchamia się np. czasochłonne zadania obliczeniowe, po czym można spokojnie wylogować się z systemu. Poniżej przykład uruchomienia graficznego edytora tekstu, z którego będziemy czasami korzystać:

```
scite &
```

Warto też wiedzieć, że znaki `*` i `?` także mają specjalne znaczenie, pozwalają mianowicie podawać nazwy plików w sposób wieloznaczny. `*` jest wzorcem pasującym do dowolnego ciągu dowolnych znaków; sama gwiazdka zostaje zatem przez shell rozwinięta do listy wszystkich plików w bieżącym katalogu. Natomiast `?` oznacza dokładnie jeden dowolny znak — zatem np. wzorec `a??` oznacza wszystkie pliki o trzyliterowych nazwach zaczynających się na literę `a`.

cd	Zmiana katalogu bieżącego	
	cd	Powrót do katalogu domowego (\$HOME)
	cd /home/guest	Przejdź do podanego katalogu
ls	Wypisanie informacji o plikach	
	ls -s plik	Podaje rozmiar pliku w blokach
	ls -la	Pełna informacja o wszystkich plikach w bieżącym katalogu
	ls -l katalog	Wyświetla listę plików w podanym podkatalogu, w 1 kolumnie
rm	Usuwanie plików (uwaga: <i>nieodwracalne</i>)	
	rm plik	Usunięcie podanego pliku
	rm f*	Usunięcie wszystkich plików o nazwie na literę f
	rm -r katalog	Usunięcie podanego katalogu z całą jego zawartością
cp	Kopiowanie plików	
	cp plik1 plik2	Tworzy kopię pliku plik1 o nazwie plik2
	cp plik(i) katalog	Jeśli ostatni argument jest katalogiem, cp kopiuje tam podane pliki
	cp * /tmp	Kopiowanie wszystkich plików z bieżącego katalogu do /tmp
	cp -r katalog1 katalog2	Kopiowanie podanego katalogu z całą jego zawartością
mv	Zmiana nazwy lub przeniesienie plików	
	mv plik1 plik2	Zmiana nazwy pliku plik1 na plik2
	mv plik katalog	Jeśli ostatni argument jest katalogiem, mv przenosi tam podane pliki
	mv * /tmp	Przeniesienie wszystkich plików z bieżącego katalogu do /tmp
mkdir	Utworzenie katalogu	
	mkdir katalog	
rmdir	Usuwanie katalogu	
	rmdir katalog	Kasuje podany katalog, o ile jest pusty (jest zatem bezpieczniejsza niż rm -r)
cat	Łączy i wypisuje dane (z plików lub standardowego wejścia)	
	cat > dane.txt	Czyta dane z klawiatury i wypisuje je; strzałka skierowuje te dane do pliku dane.txt (zakończyć należy znakiem końca pliku, czyli Ctrl-D)
	cat *.txt	Wypisuje łącznie zawartość wszystkich plików z rozszerzeniem .txt z bieżącego katalogu
chmod	Zmiana praw dostępu do plików	
	chmod a-w plik	Zabiera (-) wszystkim (a jak all) prawo pisania (w jak write) do pliku plik (staje się on plikiem tylko do odczytu)
	chmod +x plik	Nadaje plikowi prawo wykonywania (x jak eXecute) (plik będzie traktowany jak program)
	chmod ugo=rx	Ustawia prawa odczytu (r jak read) i wykonywania, ale bez prawa zapisu, jednakowe dla właściciela (u jak user), grupy (g jak group) i pozostałych (o jak others)
more	Wyświetla dane po jednej linii, lub strona po stronie	
less	Jak more, ale pozwala na przewijanie w przód i w tył	
vi	Podstawowy edytor tekstu	
nano	Podstawowy edytor tekstu	
alias	Definiuje skróty komend w shellu	
	alias dir='ls -lF'	Tworzy komendę dir, równoważną wersji podanej w apostrofach (składnia jak dla bash)
	alias	(Bez parametrów) wypisuje listę utworzonych aliasów

3 Obsługa edytora vim

Wszystkich komend edytora vi należy używać w *trybie komend*, w przeciwnym razie zostaną potraktowane jako zwykłe znaki do wpisania. Komendy vi można podzielić na dwa rodzaje: „niewidoczne”, które działają natychmiast, oraz komendy „dwukropkowe”, które widać w ostatniej linii terminala (i można je tam poprawiać), a które „zadziałają” dopiero po wciśnięciu klawisza `Enter ↵`. Te pierwsze można poprzedzić liczbą całkowitą (uwaga: także nie będzie widoczna podczas wpisywania), która jest interpretowana zwykle jako liczba powtórzeń danej komendy. Np. `10k` przeniesie kursor o dziesięć linii w górę, `20G` przeniesie kursor do linii nr 20, a `5A` spowoduje przejście do trybu dopisywania do końca bieżącej linii, jednak w momencie powrotu do trybu komend dopisany tekst zostanie powtórzony pięciokrotnie. Komendy „dwukropkowe” także mogą przyjmować argumenty liczbowe, interpretowane zwykle jako numer linii lub zakres numerów linii, do których ma się odnosić dana komenda. Numery te wpisujemy po dwukropku, ale przed literą komendy.

Podstawowe komendy	
Powrót z trybu edycji do trybu komend	<code>Esc</code>
Zapisanie i zakończenie pracy	<code>shift + [Z] + [Z]</code>
Zapis do pliku o nazwie <code>plik.txt</code>	<code>:w plik.txt</code>
Tylko zapis	<code>:w</code>
Wyjście bez zapisywania zmian	<code>:q!</code>
Wejście w tryb edycji	
Wstawianie tekstu	<code>i</code>
Dopisywanie do końca linii	<code>A</code>
Wstawianie tekstu — nowa linia powyżej kursora	<code>O</code>
Wstawianie tekstu — nowa linia poniżej kursora	<code>o</code>
Cofnięcie ostatniej zmiany	<code>u</code>
Komendy kasowania i wklejania	
Kasowanie znaku (jak Delete)	<code>x</code>
Kasowanie znaku przed kursorem (jak Backspace)	<code>X</code>
Kasowanie całej linii (jak <i>Wymij</i>)	<code>dd</code>
Zapamiętanie całej linii (jak <i>Kopiuj</i>)	<code>yy</code>
Wstawienie skasowanego tekstu (jak <i>Wklej</i>)	<code>p</code>
Zaznaczanie blokowe	
Zaznaczenie wierszy	<code>↑ + [V] + kursor</code>
Zaznaczenie bloku	<code>Ctrl + [V] + kursor</code>
Kasowanie zaznaczonego bloku (jak <i>Wymij</i>)	<code>d</code>
Zapamiętanie zaznaczonego bloku (jak <i>Kopiuj</i>)	<code>y</code>
Wstawienie skopiowanego/skasowanego tekstu (jak <i>Wklej</i>)	<code>p</code>
Poruszanie się po tekście	
W górę	<code>↑ lub [k]</code>
W dół	<code>↓ lub [j]</code>
W lewo	<code>← lub [h]</code>
W prawo	<code>→ lub [l]</code>
Na początek linii	<code>[0]</code>
Na koniec linii	<code>[\$]</code>
Przejście do linii o podanym (najpierw) numerze	<code>[G]</code>
Sprawdzenie w której linii i jakiego pliku jest kursor	<code>Ctrl + [G]</code>
Wyszukiwanie i zamiana tekstu	
Wyszukiwanie w przód	<code>/szukany tekst</code>
Wyszukiwanie w tył	<code>?szukany tekst</code>
Zamiana <code>stary</code> na <code>nowy</code> w bieżącej linii	<code>:s/stary/nowy/g</code>
Zamiana <code>stary</code> na <code>nowy</code> od linii 10 do końca pliku	<code>:10,\\$s/stary/nowy/g</code>
Zamiana każdego pierwszego wystąpienia w linii <code>stary</code> na <code>nowy</code>	<code>:1,\\$s/stary/nowy/</code>

4 Praca zdalna w trybie terminalowym

4.1 Protokół Open SSH


SSH (ang. secure shell) to standard protokołów komunikacyjnych w sieciach komputerowych TCP/IP, w architekturze klient-serwer. Protokół ten służy do terminalowego łączenia się ze zdalnymi komputerami, przy czym transfer wszelkich danych jest zaszyfrowany jednym z wybranych algorytmów kryptograficznych z kluczem publicznym (w naszym wypadku DSA lub RSA).

Protokół SSH	
<u>Komunikacja ze zdalnym hostem</u>	
połączenie z hostem	<code>ssh -Y bem.wcss.pl</code>
wykonanie polecenia na zdalnym hoście	<code>ssh bem.wcss.pl "qstat -u \\${USER}"</code>
<u>Kopiowanie plików</u>	
kopiowanie plików ...	<code>scp plik bem.wcss.pl:</code>
... pomiędzy kartotekami	<code>scp ~/kartoteka1/plik bem.wcss.pl:kartoteka2/</code>
kopiowanie kartotek do ...	<code>scp -r lokalna_kartoteka bem.wcss.pl:</code>
... i ze zdalnego hosta	<code>scp -r bem.wcss.pl:zdalna_kartoteka .</code>

Warto zwrócić uwagę na opcję `-Y` polecenia `ssh`, która pozwala na zdalne uruchamianie programów w trybie graficznym.

Aby wygenerować swój prywatny i publiczny klucz:

```
cd ~
ssh-keygen -t rsa
```

Zostaniemy zapytani o podanie frazy szyfrującej do zabezpieczenia klucza (*passphrase*). Jeśli korzystamy z bezpiecznego konta moglibyśmy nie podawać hasła (wciskając dwukrotnie ) , jednak zawsze bezpieczniej jest zabezpieczyć klucz hasłem. Można teraz skopiować klucz publiczny do zdalnego hosta:

```
cat ~/.ssh/id_rsa.pub | ssh login@bem.wcss.pl \
"cat - >> .ssh/authorized_keys"
```

W powyższym przykładzie, gdzie `login` to nazwa użytkownika, polecenie `cat` przekierowuje zawartość pliku `id_rsa.pub`, z katalogu `.ssh/` znajdującego się w katalogu domowym `~`, na standardowe wyjście. Następnie strumień ten jest przekierowany (symbol `|`) do polecenia `ssh`. Generalnie polecenie `ssh login@bem.wcss.pl` utworzyłoby po poprawnej autoryzacji połączenie terminalowe z hostem `bem.wcss.pl`. Ponieważ jednak pojawił się dodatkowy argument (polecenie w apostrofach, znak `\` pomija tylko znaczenie specjalne znaku końca linii) `ssh` otworzy polecenie terminalowe i uruchomi zdalnie polecenie w apostrofach, a po jego wykonaniu zamknie połączenie. Polecenie w apostrofach przechwytuje standardowe wejście (`cat -`) i przekierowuje je (`>>`, tu ważne są dwa znaki `>`, ponieważ jeden oznaczałby nadpisanie pliku) do pliku `.ssh/authorized_keys`. Oczywiście po uruchomieniu polecenia `ssh` zostaniemy poproszeni o hasło, jednak przy kolejnej próbie połączenia z hostem

```
ssh login@bem.wcss.pl
```

powinniśmy móc połączyć się już bez podawania hasła, chyba, że przy generacji klucza RSA zdecydowaliśmy się na zabezpieczenie go hasłem (*passphrase*). W tym ostatnim wypadku przed połączeniem powinniśmy skorzystać z polecenia `ssh-add`, przy czym hasło klucza zostanie zapamiętane przez całą długość sesji i również będziemy mogli łączyć się bez jego podawania.

5 System kolejowania PBS

PBS (ang. portable batch system) jest jednym z systemów kolejowania zadań stosowanych na serwerach obliczeniowych, zarządzający zasobami, rezerwujący i przydzielający je zadaniom, zgodnie z przyjętą polityką (zwykle na podstawie kolejności wstawienia zadania).

Podstawowe polecenia PBS

Sprawdzanie zadań w kolejce

```
wyświetl zadania w kolejkach          qstat
wyświetl zadania wybranego użytkownika qstat -u $USER
wyświetl informacje o zadaniu nr. Job_ID qstat -f Job_ID
wyświetl dostępne węzły               pbsnodes -a
```

Wstawianie zadań do kolejki

zadanie interaktywne (rezerwujące 4cpu i 4gb RAM na 6 godzin):

```
qsub -I -X -l select=1:ncpus=4:mem=4gb -l walltime=6:0:0
```

Przykład zadania rezerwującego 4cpu i 2mb RAM na 1 godzinę. Tu tworzony jest tymczasowy skrypt powłoki BASH (między wierszami zawierającymi EOF), który następnie jest przekierowany (|) do polecenia qsub. W tym przykładzie skrypt przechodzi do katalogu domowego (~) i listuje jego zawartość przekierowując standardowe wyjście i strumień błędów (>&) do pliku lista.

```
cat << EOF | qsub -r n
#!/bin/bash
#PBS -N nazwa_zadania
#PBS -l ncpus=4
#PBS -l mem=2mb
#PBS -q main
#PBS -m n
#PBS -l walltime=1:0:0
cd ~
ls -l >& lista
EOF
```

W tym wypadku wykonanie zadania jest bardzo szybkie ale jeśli szybko sprawdzimy status kolejki poleceniem

```
qstat -u $USER
```

zobaczymy następujące informacje, których wynika, że zadanie zostało wstawione do kolejki vshort (o tym, że zadanie czeka w kolejce informuje litera Q w przedostatniej kolumnie, dla uruchomionego zadania widniała by w niej litera R).

Job ID	Username	Queue	Jobname	SessID	NDS	TSK	Req'd Memory	Req'd Time	Elap S	Time
8495516.achille	rgora	vshort	nazwa_zada	--	1	4	2mb	01:00	Q	--