Zad. 0: Git, ssh

1 Tworzenie własnego zaproszenia dla powłoki `bash`

Postać zaproszenia do wprowadzania poleceń, które widoczne jest po otworzeniu terminala tekstowego kształtuje zawartość zmiennej systemowej `PS1`. Jej wartość możemy zmienić w prowadzając odpowiednie modyfikacje w pliku konfiguracyjnym `~/.bashrc`. Zazwyczaj w domyślnej zawartości tego pliku (o ile jest on automatycznie tworzony przy zakładaniu konta) znajdziemy wpis:

```
PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
```

Sekwencja `\u@\h:\w\$` powoduje, że w zaproszeniu wyświetla się najpierw login użytkownika, znak `@`, nazwa hosta oraz po znaku ':' ścieżka dostępu do aktualnego katalogu. Całość kończy się znakiem `$` np:

```
jkowalsk@panamint:~/zamp/zad$
```

Jeśli chcemy mieć wyświetloną tylko nazwę bieżącego podkatalogu oraz inną *końcówkę* zaproszenia, np. `'>'`, to wystarczy ten wpis zamienić na

```
PS1='${debian_chroot:+($debian_chroot)}\u@\h:\W> '
```

2 Zakładanie repozytorium

Opisana poniżej procedura ma charakter ogólny. Niemniej dla ustalenia uwagi załóżmy, że pracujemy na `panamincie`.

1. Jeżeli pierwszy raz korzystamy z programu `git` na danym koncie, musimy podać informacje dotyczące naszego naszego adresu e-mail oraz podać swoje imię i nazwisko, jako dane użytkownika zakładającego dane repozytorium/ia. Załóżmy, że użytkownikiem jest Jan Kowalski, a jego skrzynka e-mail to `jkowalsk@panamint.iiar.pwr.wroc.pl`. Wówczas odpowiednie polecenia powinny mieć postać:

   ```
   panamint:~> git config --global user.name "Jan Kowalski"
   panamint:~> git config --global user.email "jkowalsk@panamint.iiar.pwr.wroc.pl"
   ```

   Jeżeli chcemy się upewnić co do wyniku wydanego polecenia, to możemy je wykonać bez żadnego argumentu. Wówczas zostanie wyświetlona wartość skojarzona z danym symbolem, np.

   ```
   panamint:~> git config --global user.name
   Jan Kowalski
   panamint:~> _
   ```

2. Wybieramy sobie kartotekę w strukturze katalogu domowego. Załóżmy, że jest to katalog `$HOME/repos`. W nim będziemy przechowywać wszystkie repozytoria dla poszczególnych zadań.

   ```
   panamint:~> mkdir ~/repos
   ```

3. Dalej zakładamy repozytorium dla nowego zadania. W podkatalogu `$HOME/repos` utworzymy podkatalog `zamp/zad`, tzn. `$HOME/repos/zamp/zad`.

```
panamint:~> mkdir -p ~/repos/zamp/zad
```

4. Przechodzimy do podkatalogu `$HOME/repos/zamp/zad` i wykonujemy polecenie, które zainicjalizuje nowe repozytorium:

```
panamint:~> cd ~/repos/zamp/zad
panamint:zad> git init --bare
```

5. Tworzymy strukturę katalogu dla nowego zadania, np.

```
panamint:~> mkdir ~/zamp
panamint:~> cd ~/zamp
panamint:zamp> git clone ~/repos/zamp/zad
Cloning into 'zad'...
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

Gdy klonujemy repozytorium będąc na zewnętrzny komputer możemy to zrobić wykonując polecenie:

```
janek@moj_komp:zamp> git clone \
                     ssh://jkowalsk@panamint.iiar.pwr.wroc.pl:/home/jkowalsk/repos/zamp/zad
```

3 Dodawanie plików do lokalnego repozytorium

Utwórzmy plik w podkatalogu `zad` i dodajmy go do repozytorium.

```
panamint:zad> touch przyklad.txt
panamint:zad> git status       # zobaczym stan naszego obszaru roboczego
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

   przyklad.txt

nothing added to commit but untracked files present (use "git add" to track)

panamint:trunk> git add -v przyklad.txt
add 'przyklad.txt'
panamint:zad> git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

new file:   przyklad.txt
```

```
panamint:trunk> git commit -a -m 'Pierwszy plik do repozytorium'
[master (root-commit) 6004382] Pierwszy plik do repozytorium
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 przyklad.txt
```

4 Dodawanie plików do zdalnego repozytorium, aktualizacja

Pierwsze przesłanie do repozytorium:

```
panamint:zad> git push origin master
```

Dla kolejnych wystarczy już tylko:

```
panamint:zad> git push
```

Tak samo będzie, gdy łączymy się z zewnątrz.

Aktualizacja ze zdalnego repozytorium

```
panamint:zad> git pull
```

5 Generowanie kluczy dla ssh

Program `ssh` umożliwia wygenerowanie kluczy, które są wykorzystywane przy autoryzacji zdalnego dostępu. Pozwala to na bezpieczną autoryzację bez podawania hasła. Dzięki temu przy posługiwaniu się programem `git` z tunelowaniem poprzez `ssh` nie będzie konieczne wielokrotne wpisywanie hasła.

Należy jednak pamiętać, że automatyczna autoryzacja poprzez `ssh` *rodzi inne potencjalne zagrożenia. Jeżeli ktoś włamie się nam na konto, z którego mamy bezpieczny bezhasłowy dostęp na konta na innych serwerach, to włamywacz będzie miał również do nich dostęp. Z tego powodu ten typ dostępu powinien być możliwy tylko z konta, które jest bardzo dobrze chronione.*

Załóżmy dalej, że będziemy chcieli mieć bezpieczny i bezhasłowy dostęp do konta na panamincie z poziomu konta na diablo. Poniżej opisaną operację wykonujemy wtedy, gdy nie wygenerowaliśmy jeszcze żadnych kluczy. Jeżeli to już zostało wcześniej zrobione, to operację generacji należy pominąć i przejść do następnej.

```
diablo:.ssh> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jkowalsk/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jkowalsk/.ssh/id_rsa.
Your public key has been saved in /home/jkowalsk/.ssh/id_rsa.pub.
The key fingerprint is:
ef:20:45:d2:ba:05:fd:06:ff:1c:27:2c:e5:be:10:06 jkowalsk@diablo
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|       o         |
|      o E   .    |
```

```
|       = = +     |
|      . S B = .  |
|       + + * +   |
|      o . o +    |
|       . o . .   |
|          . .    |
+-----------------+
diablo:~> _
```

Serwery diablo i panamint mają wspólny system plików. Jednak dalsza procedura jest pokazana w sposób możliwie ogólny. Tak więc nie korzysta się z faktu, że serwery te działają na tym samym systemie plików.

Kolejna operacja maja na celu przekopiowanie pliku `id_rsa.pub` na panamint i dołączenie go do pliku `~/.ssh/authorized_keys`

```
diablo:~> scp .ssh/id_rsa.pub  panamint:.ssh/id_rsa.pub.diablo
Password:
id_rsa.pub                                    100% 1683     1.6KB/s   00:00
diablo:~> ssh panamint
Password:
Linux panamint 2.6.32-5-686-bigmem #1 SMP Fri Feb 15 16:26:23 UTC 2013 i686
Linux panamint 2.6.32-5-686-bigmem #1 SMP Tue Mar 8 22:14:55 UTC 2011 i686
*****************************************************************************
*                                                                           *
*                              P A N A M I N T^8                            *
*                                   Debian                                  *
*                                                                           *
*                              przegladarki www: opera,firefox              *
*                               klient pocztowy: alpine, mailx              *
*             graficzne edytory tekstu: gedit,gvim,emacs     *
*                                edytory tekstu: vi,vim,nano,emacs,pico     *
*                          programy inzynierskie: matlab,R                  *
*                     narzedzia programistyczne: ddd, designer              *
* narzedzie do tworzenia dokumentow tekstowych: latex                       *
*                             programy graficzne: xfig,qcad                 *
*                                                                           *
*             W przypadku problemow prosimy pisac na adres:                 *
*                      admin@amargosa.ict.pwr.wroc.pl                       *
*****************************************************************************
No mail.
Last login: Wed Oct  2 13:40:41 2013 from diablo
panamint:~> cd .ssh
panamint:.ssh> touch authorized_keys
panamint:.ssh> cat id_rsa.pub.diablo >> authorized_keys
panamint:.ssh> rm id_rsa.pub.diablo
```

Operacja zakończona. Pozostaje przetestować, że wszystko działa.

```
panamint:.ssh> exit
logout
Connection to panamint closed.
diablo:~> ssh panamint
```

```
Linux panamint 2.6.32-5-686-bigmem #1 SMP Fri Feb 15 16:26:23 UTC 2013 i686
Linux panamint 2.6.32-5-686-bigmem #1 SMP Tue Mar 8 22:14:55 UTC 2011 i686
*************************************************************************
*                                                                       *
*                             P A N A M I N T^8                         *
*                                  Debian                               *
*                                                                       *
*                             przegladarki www: opera,firefox           *
*                              klient pocztowy: alpine, mailx           *
*                    graficzne edytory tekstu: gedit,gvim,emacs         *
*                              edytory tekstu: vi,vim,nano,emacs,pico   *
*                       programy inzynierskie: matlab,R                 *
*                    narzedzia programistyczne: ddd, designer           *
* narzedzie do tworzenia dokumentow tekstowych: latex                   *
*                           programy graficzne: xfig,qcad               *
*                                                                       *
*             W przypadku problemow prosimy pisac na adres:             *
*                      admin@amargosa.ict.pwr.wroc.pl                   *
*************************************************************************
No mail.
Last login: Wed Oct  2 13:41:20 2013 from diablo
panamnint:~> _
```

Działa !!! :-)

6 Wymagania wstępne dla Etapu 1

Należy zapoznać się z dostarczonymi materiałami. Ponadto należy:

- Zaimplementować funkcję uruchamiającą preprocesor dla danego opisu działań, która wczyta i udostępni w postaci obiektu klasy `string` wspomniany opis z rozwiniętymi makrami i usuniętymi komentarzami.
- W dostarczonym zalążku należy zdefiniować obiekt klasy `LibInterface` (patrz `diagram_klas__interpreter.dia`), tak aby *zebrać* wszystkie zmienne i odwołania odnoszące się do danej biblioteki/wtyczki w jeden obiekt. Należy zauważyć, że klasa `LibInterface` przedstawiona w `diagram_klas__interpreter.dia`) jest niepełna.
- Aktualną wersję programu należy umieścić w repozytorium systemu kontroli wersji `git`.