Niby skryptowanie, ale nie tylko

Celem dzisiejszych zajęć będzie napisanie skryptu, który wykona backup całego katalogu /home do pliku user_home_databackupu.tar.gz i zapisze go w katalogu /tmp/backup. Jak ktoś wie jak to zrobić to do roboty, jak nie, to na zajęciach przejdziemy przez wszystkie elementy składowe takiego skryptu, a waszym zadaniem będzie poskładanie tego do kupy.

Pakowanie katalogu

Korzystać będziemy z programu tar. Jego nazwa pochodzi od Tape Archiver – głównie ze względu na to, że niegdyś służył do umieszczania plików na taśmach magnetycznych. Żyjemy jednak w relatywnie cywilizowanych czasach, więc zastosowanie tara się trochę zmieniło, teraz po prostu służy do pakowania wielu plików do jednego zbiorczego, tzw. archiwum. Tar pozwala przy okazji skompresować pakowane pliki przy użyciu konkretnych opcji.

```
tar [-opcje] ścieżka_docelowa źródło
```

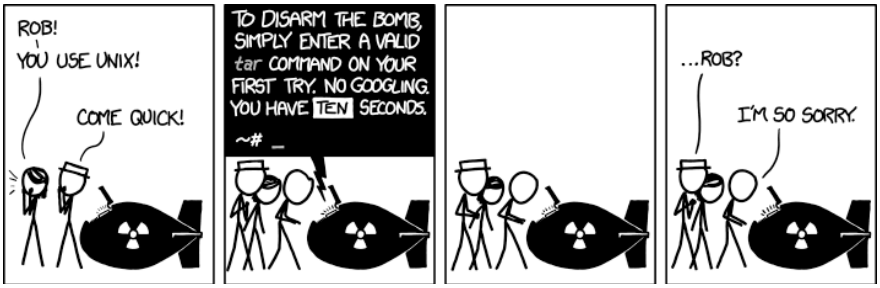
Samo wywołanie polecenia tar nic nam niestety nie da. Musimy użyć konkretnych opcji, w przeciwnym wypadku zobaczymy coś takiego:

```
michal@Mac-mini-Micha ~ % tar wyniki.tar wyniki*
Usage:
  List:    tar -tf <archive-filename>
  Extract: tar -xf <archive-filename>
  Create:  tar -cf <archive-filename> [filenames...]
  Help:    tar --help
```

I w zasadzie te informacje powinny nam wystarczyć.

```
-f – określa nazwę pliku tar
-c – tworzy plik tar
-x – rozpakowuje plik tar
-v – listuje użyte pliki
-z – kompresuje pliki gzipem do formatu tar.gz
-t – wyświetla zawartość archiwum
```

Osobiście korzystam w 90% z kombinacji -xvzf oraz -cvzf. I tak nikt nigdy tego nie pamięta



Nazwa użytkownika

Generalnie to nie jest żadna filozofia, bo nazwę użytkownik zwróci nam polecenie

```
whoami
```

Ale już bardziej skomplikowanym trikiem może być stworzenie pliku, który będzie zawierał naszą nazwę użytkownika. Najrozsądniejszym wydaje się przypisanie naszej nazwy użytkownika do zmiennej. Wtedy taka kombinacja poleceń

```
user=`whoami`
touch $user
```

Faktycznie zwróci nam to, czego oczekiwaliśmy, czyli stworzy plik o naszej nazwie użytkownika. Ale chcielibyśmy, żeby nasz plik nazywał się np. ‘michal_home_12-11-2021’. W takim wypadku polecenie

```
touch $user_home_jakastamdata
```

nie ma sensu, bo shell będzie oczekiwał zmiennej $user_home_jakastamdata. W takiej sytuacji możemy zamknąć nazwę zmiennej w nawiasy klamrowe. To jest tzw. Parameter Expansion - zupełnie niepowiązane z tym tematem, zaawansowana wiedza, której nigdy w życiu nie wykorzystałem. Na nasze potrzeby wystarczy wiedzieć, że zamknięcie zmiennej w klamerki pozwoli nam utworzyć ze zmiennej fragment stringa

```
touch ${user}_jajco
```

stworzy nam plik michal_jajco.

Data

```
michal@Mac-mini-Micha ~ % cal
   Listopad 2021
nd po wt śr cz pt so
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Zaczniemy od polecenia cal. Pokazuje datę. Znaczy ma wiele opcji, ale koncept jest dość prosty i nie po to tu dziś przyszliśmy. Nasz plik ma posiadać datę utworzenia backupu, czyli potrzebujemy raczej daty w formie zmiennych. I do tego posłuży nam polecenie date. Ono służy do wyświetlania bądź ustawiania czasu.

```
date +%Y
```

zwróci bieżący rok. Znak + oznacza zastosowanie formatu wypisu daty, w którym pod konkretnymi kodami znajdują się konkretne informacje. Używamy standardu ISO 8601, a konkretne elementy daty zastępowane są odpowiednimi tokenami.

```
michal@Mac-mini-Micha ~ % date +%Y
2021
```

Tokeny możemy łączyć w stringi zasadniczo dowolnie, dla przykładu:

```
michal@Mac-mini-Micha ~ % date '+dziś %d dzień miesiąca %B roku pańskiego %Y'
dziś 12 dzień miesiąca listopada roku pańskiego 2021
```

Albo dla cywilizowanych ludzi:

```
michal@Mac-mini-Micha ~ % date +%d.%m.%y
12.11.21
```

Po wszystkie możliwe opcje dotyczące formatu zapraszam do manuala, o tutaj. Have Fun.

Śmietnik

W linuksach jest takie magiczne miejsce, które jest czarną dziurą. Przyjmie wszystko, zwróci nic. Nazywany różnie, *sink, czarna dziura, Dave Null,* służy do jednego – przemielenia i usunięcia wszystkiego, co się do niego wrzuci. Może służyć np. jako generator pustej zawartości dla plików, a zwykle, jako genialni deweloperzy przekierowujemy do niego STDERR. Przecież wiemy, że działa, nie?

```
2> /dev/null
```

Skrypt finałowy

Przypomnę, zadanie polega na napisaniu skryptu, który weźmie nasz cały katalog domowy i spakuje go do pliku ‘tmp/backup/[nazwa_użytkownika]_home_[data_w_dowolnym_formacie].tar.gz’. Np. mój będzie się nazywał ‘michal_home_2021-11-12’.