du Bytes Sünder

Situation: Habe ich noch Platz auf meiner Harddisk?
Ob sich diese Frage heute noch stellt? Bei Terra von Bytes Platz auf der Harddisk. 😉
Doch wieso dauert mein Backup solange?

Was sind die grössten Speicherplatz-Sünder auf meiner Harddisk?
Und wo liegen sie?

Manchmal wäre es durchaus nützlich, zu wissen, wieviel Platz durch was belegt ist.

Sie lernen:

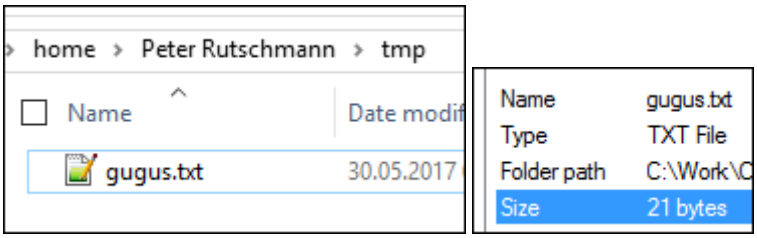
- Anwenden von *du*.
- Vertiefen von Ausgabe-Umleitung
- Anwenden von Parametern bei Scripts

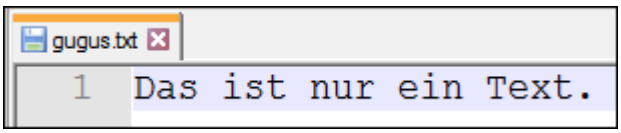
Aufgabe

Untersuchen Sie die Möglichkeiten von *du* mit Hilfe der Hilfe und den Man-Pages.

- Was macht *du,* wenn man es ohne Optionen oder Parameter aufruft?

  Erstellen Sie ein Verzeichnis und im Verzeichnis eine Datei.






```
Peter Rutschmann@Odysseus ~/tmp
$ du
1       .
```

  Was bedeutet die 1 und was der Punkt?
  Wieso steht 1 und nicht 21?

  Probieren Sie mal diesen Befehl aus:

```
$ du --apparent-size --block-size=1
21      .
```

- Schlagen Sie nach was die Optionen bedeuten.
  - - apparent-size
  - - block-size=1

- Blöcke oder Bytes, etwas verwirrlich…

  Wie lauten die Optionen von *du*, damit die Grösse aller Dateien und Verzeichnisse in Bytes aufgelistet werden?

- Wie lauten die Optionen von *du*, damit die „Summer über alles“ ausgegeben wird?
  (Also nicht jede Datei und Verzeichnis einzeln…)

Aufgabe mit exec

Wir haben uns bisher mit ein paar Aspekten von *du* vertraut gemacht.

Doch nun zu unserem eigentlichen Ziel:

- „Wir möchten unsere Speicher-Sünder herausfinden.“
- Sie müssen ein Script schreiben, dass die Grösse aller Dateien (und nur der Dateien) sortiert ausgibt. Die Grössten sollen zuerst erscheinen.

➔ Problem: Nun kann ich zwar mit *du* «nur» die Verzeichnisse auflisten, jedoch nicht die Dateien alleine…

Erste Idee:

- Wir verketten zwei Utilities mit einer Pipe

  find soll alle Dateien finden → Pipe → *du* soll die Grösse bestimmen.

```
$ find . -type f
./blabla.txt
./gugus.txt
./subdir/Text1.txt
./subdir/Text2.txt
```

➔ und mit Pipe ➔

```
$ find . -type f | du -b
1128    ./subdir
2438    .
```

Uuups, das Resultat zeigt wieder alle Verzeichnisse.
Hmm, die Pipe scheint nicht zu funktionieren.

➔ Merke: Es gibt Utilities, die den Input über eine Pipe nicht verwenden.
du ist ein solches… eigentlich logisch, da *du* auf Verzeichnisse und Dateien zugreift…

Zweite Idee:

- *du* kann auch die Grösse von definierten Dateien bestimmen.

```
$ du gugus.txt -b
21      gugus.txt
```

… wenn ich nun mit *find* eine Liste erstelle und die *du* als Parameter übergeben könnte…

Doch wie bekomme ich den *Output* eines Utilities als *Parameter* eines anderen Utilities hin?

Die Antwort darauf finden wir bei *find*:

Lesen Sie in den Man-Pages zu find die Einträge «**-exec command ;**» und «**-exec command {} +**»

- Und dann suchen Sie mit Google nach «getting size with du of files only»

  Lösung gefunden?? 😊
  Ausprobieren!!

```
-exec command ;
       Execute  command;  true  if  0  status is returned.  All following arguments to find are
       taken to be arguments to the command until an argument consisting of `;' is encountered.
       The  string  `{}'  is  replaced  by  the current file name being processed everywhere it
       occurs in the arguments to the command, not just in arguments where it is alone,  as  in
       some  versions  of  find.   Both of these constructions might need to be escaped (with a
       `\') or quoted to protect them from expansion by the shell.  See  the  EXAMPLES  section
       for examples of the use of the -exec option.  The specified command is run once for each
       matched file.  The command is executed in the starting directory.   There  are  unavoid-
       able  security problems surrounding use of the -exec action; you should use the -execdir
       option instead.

-exec command {} +
       This variant of the -exec action runs the specified command on the selected  files,  but
       the  command  line  is  built by appending each selected file name at the end; the total
       number of invocations of the command will be much less than the number of matched files.
       The  command  line  is  built  in much the same way that xargs builds its command lines.
       Only one instance of `{}' is allowed within the command.  The command is executed in the
       starting  directory.  If find encounters an error, this can sometimes cause an immediate
       exit, so some pending commands may not be run at all.   This  variant  of  -exec  always
       returns true.
```

Script duBytesSuender.sh

- Schreiben Sie mit dem erlangten Wissen nun das Script, dass zuerst die Summe der Grösse aller Dateien in Bytes ausgibt. Und danach die drei grössten «Speichersünder» auflistet.

```
$ ./duBytesSuender.sh
Speicherplatzbedarf total:
4503

Die groessten Suender:
1316    ./duBytesSuenderParam.sh
1289    ./blabla.txt
1107    ./subdir/Text2.txt
```

- Wir brauchen:
  - du
  - cut (damit hinter 4503 kein Punkt erscheint…. Hilfe zu cut sagt wie man das macht…)
  - find mit -exec
  - head (damit nur die grössten 3 Files ausgegeben werden.

Script duBytesSuenderParam.sh

- Erweitern Sie Ihr Script, so dass ich das Verzeichnis als Parameter übergeben werden kann.

```
$ ./duBytesSuenderParam.sh subdir
Speicherplatzbedarf von 'subdir' betraegt in Bytes total:
1128

Die groessten Suender:
1107    subdir/Text2.txt
21      subdir/Text1.txt
```

- Gibt man keine Parameter ein, so gibt das Script eine Anleitung aus.

```
$ ./duBytesSuenderParam.sh
Usage ./duBytesSuenderParam.sh [directroy]
Dieses Script listet ab einem Verzeichnis rekursiv
die Summe des Speicherbedarfs auf.
Sodann noch die drei groessten Dateien.

Sie muessen beim Aufruf des Scripts ./duBytesSuenderParam.sh
den Name des Verzeichnises mitgeben.
```