Welcome gks-017
!

GridKA 2012 - IO/Data Management-Exercises Material

Exercises:

[back to top]

1.1 Time Measurements - exercise 1

Measure the execution time of the command

```
sleep 1
```

in the bash shell! This might sound **silly** but you will get a feeling that one can do big mistakes in trivial measurements. The answer is the command you issue in a bash prompt!

Hint

Use the time command in front of **sleep 1**.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.1 Time Measurements - exercise 2

We are going to do IO measurements. Which type(s) of time measurements are more relevant in the context of IO measuremts:

1. REAL (real-time)
2. SYS (system time)
3. CPU (user cpu time)

You can answer REAL SYS CPU or any combination.

(answered)

Hint

Many IO operations are reflected in an increase of system time - therefore it is interesting to measure the system time. In most cases we want to calculate IO rates like MB/s. What is **per second** in that case?

Solution

Certainly we need to do realtime measurements to compute IO rates. Nevertheless system time is interesting because the number of IO operations and IO bandwidth the OS can do is limited and these are reflected by system time measurements.

Answer provided:

done

[back to top]

1.1 Time Measurements - exercise 3

Why is the real-time of the process not exactly 1 second?

1. The **time** command itself consumes a considerable amount of realtime and changes the measurement result
2. The OS has to find the executable and libraries to run sleep - this adds few ms to the realtime
3. That is wrong - most of the time we measure exactly one second!

Hint

Try to think, how this time measurment is made in the OS. **Attention:** there are two time commands. If you run the bash shell you get the one implemented in bash, but there is also the GNU time command /usr/bin/time which shows the time with less resolution but can give you even more information than just cpu and real-time measurements. See the man page ...

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.1 Time Measurements - exercise 4

Give a lower boundary in **ms** for the precision of a realtime measurement using time in the bash shell!

Hint

What is the smallest time you could measure? Do not consider the systematic shift of few **ms** you measure!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.1 Time Measurements - exercise 5

What is the nature of the error which is given by the startup phase of the program we want to measure? How do you call such an error?

Hint

Errors can be of statistic or systematic nature. Moreover it can be proportional or independent of the value you measure, therefore your answer should contain some of these key words:

- statistic
- systematic
- proportional

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.1 Time Measurements - exercise 6

How would you do a more precise realtime measurement if you implement the sleep command yourself as a compiled program? Here you cannot give a trivial answer. Try to imagine, then get a confirmation or not by the hint and solution and you terminate the exercise by typing **done**.

Hint

There is a system call to provide a **sleep 1** and there are system calls to measure times with the precision of **µs** or **ns**! How does your program looks like?

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.2 top/vmstat - exercise 1

Identify the process consuming most of your physical memory sorting the output of the **top** command by memory consumption! How do you do that?

Hint

Execute **top**, then press **M**. Just in case: you can leave top with **q**.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or*

some text explaining your observation. Your response is not automatically evaluated

Finish exercise

[back to top]

1.2 top/vmstat - exercise 2

Switch back to the process CPU view and change the update frequency of the top window to 2 Hz! Which keys do you press?

Hint

You can always get information via **man top**. You sort again by CPU consumption via **P** and you can set the update frequency by pressing **d** and the time in seconds between updates.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.2 top/vmstat - exercise 3

Switch the top window to 1000Hz update rate. Get familiar with the command **vmstat** and run it in a second window with a 1Hz update frequency. How much CPU time is used by your **top** window?

Hint

You start top, then press the key sequence **s0.001[ENTER]**. You run vmstat with 1Hz like **vmstat -n 1**.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.3 iostat/dd/strace - exercise 1

Run a **vmstat** and an **iostat** (use -x) command with 1 Hz update frequency in separate windows. Use now the **dd** command to create a 1MB file /tmp/1MB. Use a blocksize of 1 byte and repeat the measurement few times to verify the result. What is the IO rate you measure? Put a number in kB/s as answer.

Hint

You have to define the inputfile and outputfile via arguments. Use **/dev/zero** as input device, the local file as output device. Then you set the blocksize with additonal arguments: to write 1MB with 1 byte blocks you add the options **bs=1 count=1048576**.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.3 iostat/dd/strace - exercise 2

Look at the output of vmstat and iostat during the previous exercise. How is it possible that it takes close to a second to write a small file? Shouldn't a harddisk be much faster? What is the limiting factor? Consider to use **strace** to inspect how our **dd** command is implemented. Where is the bottleneck?

Hint

The meaning of **bs=1** is revealed when you run **strace dd ...**

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.3 iostat/dd/strace - exercise 3

Inspect system calls using the strace command in front of the yes command. Compare the result when redirecting STDOUT to /dev/null (or a file). What is the difference? What happens during shell redirection?

Hint

To redirect STDOUT to a file you add to the command **> /tmp/myfile**. Compare the size of write system calls!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.3 iostat/dd/strace - exercise 4

Run the dd command creating a 10MB file for various blocksize: 1, 8, 16, 256, 4096, 256k, 1M. Draw **IO rate in MB/s vs blocksize**! When finished, give **done** as answer.

Hint

You just need to vary the **bs=..** parameter and make sure that the product **blocksize*count**=1MB.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.3 iostat/dd/strace - exercise 5

Run the **yes** command and redirect the output into the file **/tmp/yes**. Inspect the vmstat and iostat windows while it runs and after it finishes. You can also try instead another tool called **dstat** which has simpler user interface. Which cache strategy is used by the OS when you write files:

1. no cache
2. write-through cache
3. write-back cache
   Cleanup the potentially huge file you created.

Hint

Pay attention to the block output value and the timing when the output to disk starts.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.4 cp - exercise 1

Use the strace command to inspect only open,read,write,close,stat calls for the following commands:

1. cp /etc/goup /tmp/group
2. cp /usr/bin/vim /tmp/vim Which IO pattern is used to do the copy e.g. what is the copy unit?

Hint

The strace command restricting to IO operations is **strace -e file,read,write**. Try to find the return value of the open command of the input file. This is the file descriptor used for that file. Then look for read calls on that file descriptor and the length specified for the read call.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.4 cp - exercise 2

Give the realtime overhead in percent (answer just the plain number) comparing the second copy command in the previous exercise induced by running strace.

Hint

Measure the two numbers and express the overhead in percent e.g. if you measure something like 5s compared to 1s the overhead is 400 percent.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

1.5 strace - exercise 1

strace is a very valuable tool to debug IO related problems in executables. Moreover it is usefull to understand binary programs without having source code at hand (e.g. you can attach strace to a running program like you do with a debugger).
Try to investigate with strace and top or vmstat what the execution of the mysterious program **/data/dm/bin/bash** does. Be careful not to loose track of it! Write some keywords of your observation as the answer (the answer is not evaluated automatically).

Hint

Run the command immedeatly with strace, you will understand that it is quite impossible to attach to it afterwards. Add also the **-f** switch to follow forks. Look for connect statements, file open and forks. Best practice is to redirect the output into a file and then step through it with an editor like vim,emacs or pico.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 1

DISK WRITING
As you learned in the first exercise you can write a file F of length N with blocksize B containing only zeros like that:

```
dd if=/dev/zero of=F bs=B count=N divided by B
e.g. dd if=/dev/zero of=/tmp/1MB bs=32k count=64
```

DISK READING
Can you imagine how to read back the existing file /tmp/1MB using the dd command dumping the output into the **/dev/null** device? Can you do the same with a **cat** command? Investigate quickly if there is a difference in the IO pattern using strace on both commands. Just put **done** as the answer when you finished the exercise.

Hint

You can specify a file for reading with the **if=FILENAME** option. You can dump a file to STDOUT with the **cat** command and redirect the output to **/dev/null**.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 2

Write sequential files using dd with 4k IO blocks with a size of 1MB, 10MB, 100MB, 1GB, 4GB into the /tmp/ directory with their size as filename. Does the real-time scale linear? Explain what you observe!

Hint

With every file we increase the size by a factor of ten. Linear scaling means, that also the real-time increases by a factor of 10 e.g. the performance stays the same.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 3

Run first the command **dropcache**. Measure the time to read back the previously generated 5 files **two** times each using 4k blocksize with **dd**! How would you explain a large variation from the 1st to 2nd execution? Are all results compatible with the performance of a single hard disk? (The answer is not evaluated automatically).

Hint

How does the file size impact the performance? How big is your machine memory?

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 4

There is a way in Linux to write directly from memory to a device using the DMA and the other way around.
You can specify this so called **direct IO** by adding **oflag=direct** to the dd command.
Write a 10MB file using the following blocksizes: 1k,4k,16k,65k,256k,1M. Explain your observation: can it be an effect of the disk, cache, system call performance?
You can verify that it works also with 512 bytes, but you can see, that it does not work

with smaller numbers like 256 or with numbers like 1000,4000 aso.!

Do you have an idea what has to be guaranteed for direct IO to work? Write your answer, it is not automatically evaluated.

Hint

You can find the info about direct IO doing **man 2 open** and read what is written for the O_DIRECT flag.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 5

What is the sequential read performance of your harddisk based on a measurement reading a 1 GB file? Answer with the rate as a number in MB/s (skipping the unit).

Hint

To avoid cache effects we can measure with direct IO and large blocks (1M). If you want to use direct IO for reading you specify **iflag=direct**. An alternative is to flush the buffer cache and read the file without direct IO. As root (unfortunately you are not!) you can flush the OS caches in Linux by doing:

```
sync && echo 3 > /proc/sys/vm/drop_caches
```

As a user you can also flush the cache by writing a new file bigger than the physical memory. On the exercise machines there is a special command installed to clean the OS cache from a user account (which we used already before): dropcache

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 6

What is the sequential write performance of your harddisk based on a the measurement writing a 1 GB file with dd? Answer with the rate as a number in MB/s.

Hint

To avoid cache effects we can measure with direct IO and large blocks (1M). If you want to use direct IO for writing you specify **oflag=direct**

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.1 Performance-Monitoring-Measurement-Caching - exercise 7

Write a program that reads 1.000 randomly chosen bytes in one of your previously created 1 GB files. Calculate the average seek time of your hard drive and write the number in ms as answer. Additionally compute the realtime ratio running the program with the file uncached and cached (not considered in the answer). Hint: Take care not to be fooled by the buffer cache in case you repeat a measurement with the same **random** blocks. If you don't use a new random seed with each program start you re-read always the same blocks!

Hint

The logic to implement is:

```
open file
do 1000 times {
 choose random offset within size of file
 read a single byte
}
close file
```

Measure the execution time of you program with the **time** command which gives you approximatly the time to do 1000 seeks.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.2 IO Prioritization - exercise 1

Modern unix kernel have the option to give a priority value for IO to threads or processes. The command line interface to this functionality is **ionice**. This prioritization implements several algorithms. You can read details doing **man ionice**.

Now run in two windows two writers at the same time writing to different files with 1MB blocksize, direct IO and 1GB file size. Both writers should run with **ionice** and the best effort alogrithm. First run them both with the same priority value (0-7) and verify if there is a fair sharing of IO between both processes.

Afterwards run one with the lowest and one with the highest priority and see how the sharing of IO between both processes is. Write your observations into the answer (it is not automatically evaluated).

Hint

To run a dd process with **ionice** and best effort algorithm you execute:

```
ionice -c2 -n0 dd ...
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.3 IO Readahead - exercise 1

The following program reads a file in 4k chunks from beginning to end. Create a 1 GB file, clean the buffer cache and measure the time it takes for execution.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <unistd.h>
#include <sys/fcntl.h>

int main(int argc, char* argv[])
{
  int fd ;
  char* buffer = (char*) malloc(1024*1024); // malloc a 1M buffer

  if (!buffer) exit(-1);  // check it is malloced
  if (!argv[1]) exit(-1); // check we have a file name

  if ( (fd=open(argv[1],0,0)) ) { // open file
    size_t nread=0;
    off_t offset=0; // start at off set 0
    do {
      nread = pread(fd, buffer, 4096, offset); // read 4k at offset
      if (nread>0) {
        offset += nread; // step to the next 4k offset
      }
    } while ( nread > 0);// terminate when we receive less than requested
  }
}
```

Now modify the program to do the read loop backwards. Clean the buffer cache and measure again the time it takes. How do you explain the different execution time?

Hint

```
Start with the highest possible offset and step 4k back in each loop.
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.3 IO Buffer Cache - Disable the buffer cache - exercise 2

Use the base program of 2.3.1 and add a posix_fadvise function to instruct the buffer cache not to cache the read pages. Clean the buffer cache and re-run the program several times. The performance should not change between first and second run.

Hint

```
Use  man posix_fadvise
```

Show solution

Answer

Provide an answer for this exercise. You can just terminate the exercise by typing

done or some text explaining your observation. Your response is not automatically evaluated

Finish exercise

[back to top]

2.3 IO Buffer Cache - Advise read-ahead - exercise 3

Modify the base program of 2.3.1 to read 1Mb at each offset position 0,10M,20M,30M .... 990M. To emulate some processing of the data read we add 100ms processing time per read using **usleep 100000**. Measure the execution time with clean buffer cache. Before we call the processing (usleep) we give the OS a hint for the next read we expect using posix_advise. Does it help to improve the real-time of the program? Does it compensate all the extra IO time?

Hint

```
Use the POSIX_FADV_WILLNEED flag to give a hint to the OS.
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

2.3 IO Buffer Cache - Advise read-ahead - exercise 4

Modify the previous program and run the **posix_fadvise** in a forked child. Measure the time with cleaned buffer cache. Do not forget to call wait to collect the child process.

Hint

The structure of the fork call branch is:

```
if (!fork()) {
  // => advise
  exit(0);
 }
usleep
wait
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 1

See handout.

Hint

Call the sha1string library function!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 2

See handout.

Hint

Try

```
h8d 0
h8d 1
...
h8d a
h8d b
h8d f
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 3

See handout.

Hint

I cannot give you more hints!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 4

See handout.

Hint

```
text="123"
echo ${text:0:1}
1
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 5

See handout.

Hint

Assign the output of

```
"h8d $hashkey"
```

to the variable hashvalue!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 6

See handout.

Hint

Call the upload function with the proper arguments! Read the function documentation!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 7

See handout.

Hint

Call the download function with the proper arguments! Read the function documentation!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 8

See handout.

Hint

Call the list function with the proper arguments! Read the function documentation!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 9

See handout.

Hint

Call the delete function with the proper arguments! Read the function documentation!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 10

See handout.

Hint

To upload all files do:

```
for name in `find /data/dm/cloudfiles/ -type f`; do cloud_upload ... ; done
```

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 11

See handout.

Hint

The function is very similiar to the upload function!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 12

See handout.

Hint

Just call the list function for all hash values!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 13

See handout.

Hint

Just call the delete function with the proper arguments. Read the function documentation!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 14

See handout.

Hint

If you do not know enough of bash skip this and open the solution!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 15

See handout.

Hint

Just dump your bucket file with cat on the screen!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.1 Cloud - exercise 16

See handout.

Hint

If you do not know enough of bash skip this and open the solution!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.2 Cloud - exercise 1

See handout.

Hint

Imagine your table being a ring ...

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.2 Cloud - exercise 2

See handout.

Hint

Just add two more upload commands with the correct hash values!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.3 Cloud - exercise 1

See handout.

Hint

Use the low level delete method with the right hash value ... the primary location is located at hash value 1!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.3 Cloud - exercise 2

See handout.

Hint

Just try the alternative locations if the return of the download command is not 0!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.3 Cloud - exercise 3

See handout.

Hint

We have seen that the additional replicas help to balance the file distributions! Should we select then only the primary replica?

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.4 Cloud - exercise 1

See handout.

Hint

You can copy the h8d function and adjust the formula!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.4 Cloud - exercise 2

See handout.

Hint

List each server using the list command and use the file name list.

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

[back to top]

3.5 Cloud - exercise 1

See handout.

Hint

No more hint!

Show solution

Answer

*Provide an answer for this exercise. You can just terminate the exercise by typing **done** or some text explaining your observation. Your response is not automatically evaluated*

Finish exercise

If you have any question or comments you can email me: Andreas.Joachim.Peters@cern.ch