Climate data processing for climate resilience

Tajikistan and Kyrgyzstan

Data access, processing and methodological concepts

Webinar
17-11.2020 27.11.2020

On behalf of the GIZ- project:
Technology based adaptation to climate change in rural areas of Tajikistan and Kyrgyzstan

info@nilshempelmann.de

Suggestions :

- **Mute your microphone**
- **Select your Language**
- **You can use Chat function to ask questions and give comments**
- **Rise your hand when you would like to speak**

Objectives of the training are focusing on the following topics:

- Introduction to sustainable development
- General overview to scenarios of change concepts,
- General overview on Climate Change related Geodata,
- Future projections of Climate Change in the countries
- Model chain from global climate change to local impact for different sectors
- Options for monitoring of Climate developments and impact,
- Databases for disaster risk reduction and disaster control,
- Climate service information systems to generate climate information on demand.

Week 1 :

Day		Theroie	Hands On
1.	Tu 17.11	Introduction, Expectations	Getting started with Linux exploring the VM
2	We 18.11	SDG Concepts, SDG13, Climate Action, Institutions Scenarios of Change	Country strategies Knowing netCDF files Which data are needed? Which climate infos are needed
3	Th 19.11	Differnt Climate Data Families Which data for what	Climate Model Data Plotting in Phyton
4	Fr 20.11	Disaster Risk Reduction	Sat-Data in QGIS / SNAP

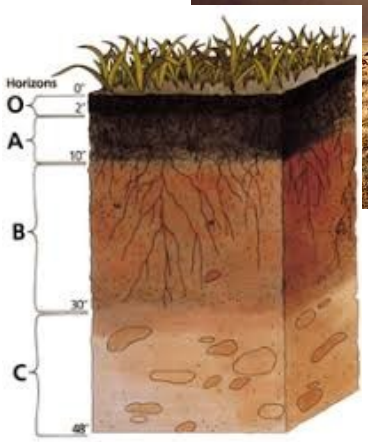
Week 2 :

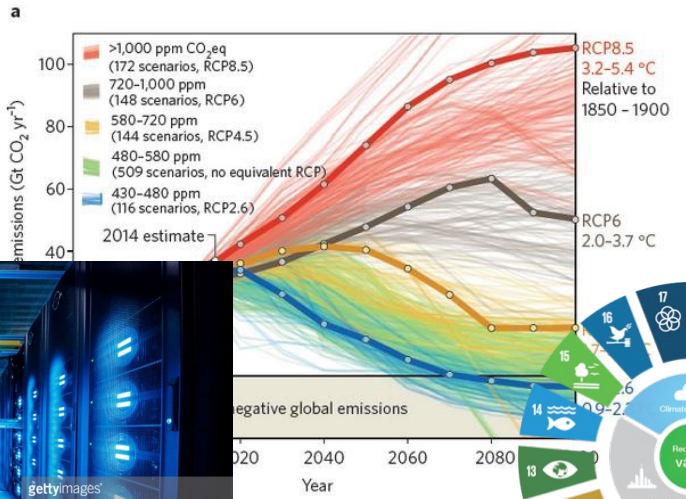
Day		Theroie	Hands On
5	Mo 23.11	Data Archives and their Access	Acess to ESGF
6	Tu 24.11	Big Data Handling Server-Side data processing	Submit a Job on a Cloud
7	We 25.11	Interoperabillity Database design	Operational FAIR Climate Service
8	Th 26.11	Climate Services Information Systems	How should a Data Center be designed
9	Fr. 27.11	Optional presentation of participants course projects	

Introduction to each other :

Dr. Nils Hempelmann

Johannes Gutenberg University Mainz 2007-2010	Doctor of natural sciences Department of Geography Grade: very good
Philippes University Marburg 2000-2006	Diplom Department of Geography Grade: good (1.5)





info@nilshempelmann.de

The Participants:

https://docs.google.com/document/d/1i6hvZ483M2oa7yzRaM1H0dEKQDGq4OvKRijjySISF7o/edit#heading=h.tyjcwt

info@nilshempelmann.de

Get Started

Launch the Virtual Machine

- Set keyboard and language settings
- Explore folders and files
- Open a terminal
- Open a climate data file with panoply

Shells

(Terminal / Console)

- Thompson-Shell osh
- Bourne-Shell sh
- C-Shell **csh**
- Job-Control-Shell jsh
- Korn Shell **ksh**
- Public-Domain-Korn-Shell pdksh
- Bourne-Again-Shell **bash**
- TENEX-C-Shell tcsh
- Z-Shell zsh
- Almquist-Shell ash
- Debian-Almquist-Shell dash
- ….

Useful commands:

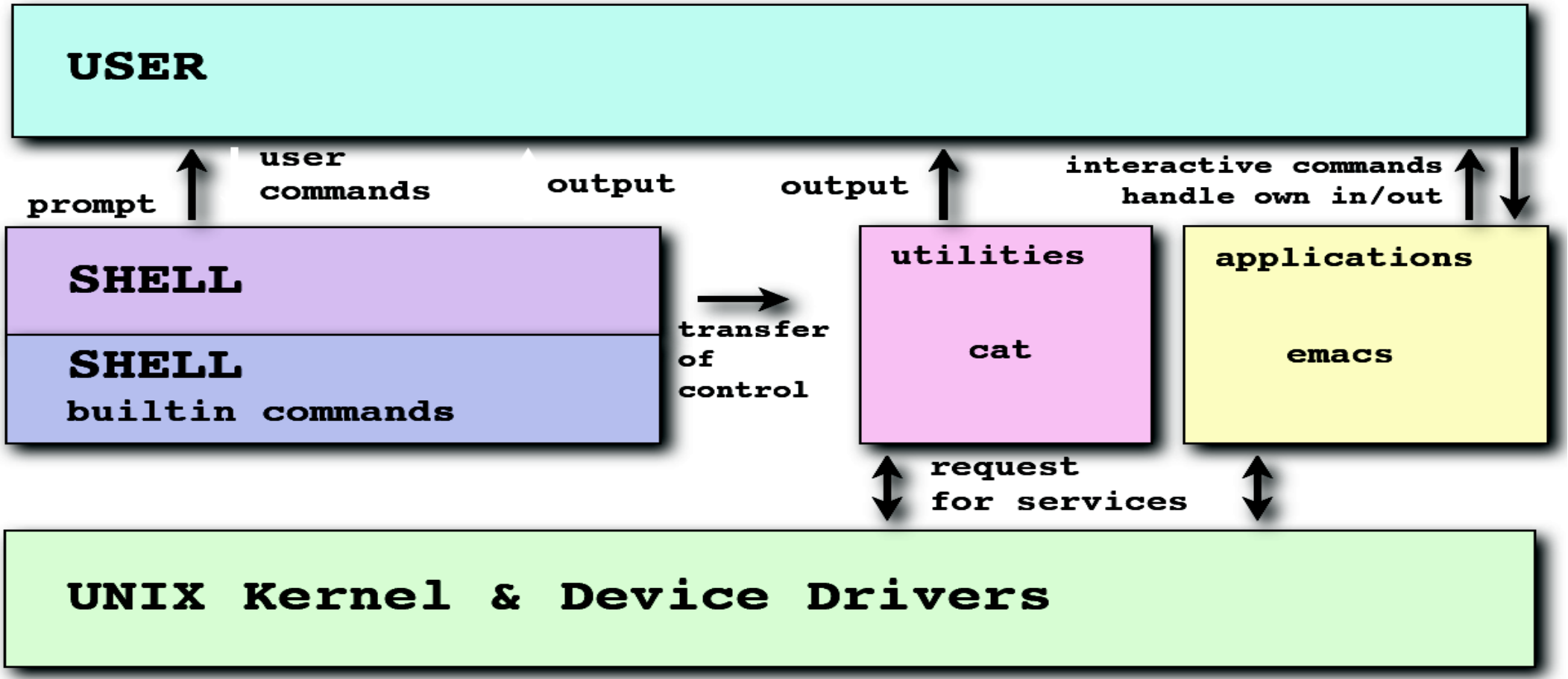
```
#!/bin/bash
echo $0
echo $SHELL
echo $PATH

#to change the shell just type the name
bash
csh

who
date
cd
mkdir
pwd
ls -al
history

man ls


exit
```


USER
prompt
user commands
output
output
interactive commands handle own in/out
SHELL
SHELL
builtin commands
transfer of control
utilities
cat
applications
emacs
request for services
UNIX Kernel & Device Drivers

Paths:

Unix searches for commands, scripts and programms in directories thar are difined in the **PATH variable**.

It is possible to define and export own pathes

Useful commands:

```
# csh
set PYTHONPATH = “/path/to/python“

# bash
export PYTHONPATH=“/path/to/python“
```

Wildcards

Wildcards allow to have access to more then one file with one command.
They are substitutes for none, one or more character.

*	for none or more character
?	for exactly one character
[n-m]	for exactly one character out of n-m,
[n,m]	for exactly two characters n and m,
^	negation
{text1,text2,..}	strings separated by commas

Useful commands:

```
rm *.txt
cp car? $HOME
rm [d,e]
more file?
more file*
ls -l ^file*
```

local Variables

Variables contain informations.

We distinguish between **predefined** shell variables (system variables) and **user defined** variables.

local variables, only callable during the process they are created in.

Useful commands:

#!/bin/bash
variable=value
variable="va lue"

#!/bin/csh
set variable = value
set variable = "va lue"

create empty variables
set variable

delet varables
unset variable1 variable2

list of all defined variables
set

call the content
$variable

global Variables

Global variables created with

setenv (csh)

env (bash)

are available in all child processes.

Variables can be:

- String variables
- Integer variables
- Constant variables
- Array variables

Useful commands:

```
#!/bin/bash
#set a global variable
env variable value

#!/bin/csh
#set a global variable
setenv variable value

# delete a global variable
unsetenv variable

# list of all defined global variables
#!/bin/bash
env
#!/bin/csh
setenv
```

./scripts/mysystem.sh

```
#!/bin/bash
clear
echo "This is information provided by mysystem.sh. Program starts now."
echo "Hello, $USER"

echo
echo "Today's date is `date`, this is week `date +"%V"`."
echo

echo "These users are currently connected:"
w | cut -d " " -f 1 - | grep -v USER | sort -u
echo

echo "This is `uname -s` running on a `uname -m` processor."
echo

echo "This is the uptime information:"
```

./scripts/positional.sh

```
#!/bin/bash
# positional.sh
# This script reads 3 positional parameters and prints
them out.
POSPAR1="$1"
POSPAR2="$2"
POSPAR3="$3"

echo "$1 is the first positional parameter, \$1."
echo "$2 is the second positional parameter, \$2."
echo "$3 is the third positional parameter, \$3."
echo
```

Calculation (in bash)

Operator	Meaning
VAR++ and VAR--	variable post-increment and post-decrement
++VAR and --VAR	variable pre-increment and pre-decrement
- and +	unary minus and plus
! and ~	logical and bitwise negation
**	exponentiation
*, / and %	multiplication, division, remainder
+ and -	addition, subtraction
<< and >>	left and right bitwise shifts
<=, >=, < and >	comparison operators
== and !=	equality and inequality
&	bitwise AND
^	bitwise exclusive OR
|	bitwise OR
&&	logical AND
||	logical OR
expr ? expr : expr	conditional evaluation
=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^= and |=	assignments
,	separator between expressions

Conditional statment -- If

Primary	Meaning
[-a FILE]	True if FILE exists.
[-b FILE]	True if FILE exists and is a block-special file.
[-c FILE]	True if FILE exists and is a character-special file.
[-d FILE]	True if FILE exists and is a directory.
[-e FILE]	True if FILE exists.
[-f FILE]	True if FILE exists and is a regular file.
[-g FILE]	True if FILE exists and its SGID bit is set.
[-h FILE]	True if FILE exists and is a symbolic link.
[-k FILE]	True if FILE exists and its sticky bit is set.
[-p FILE]	True if FILE exists and is a named pipe (FIFO).
[-r FILE]	True if FILE exists and is readable.
[-s FILE]	True if FILE exists and has a size greater than zero.
[-t FD]	True if file descriptor FD is open and refers to a terminal.
[-u FILE]	True if FILE exists and its SUID (set user ID) bit is set.
[-w FILE]	True if FILE exists and is writable.
[-x FILE]	True if FILE exists and is executable.
[-O FILE]	True if FILE exists and is owned by the effective user ID.
[-G FILE]	True if FILE exists and is owned by the effective group ID.
[-L FILE]	True if FILE exists and is a symbolic link.
[-N FILE]	True if FILE exists and has been modified since it was last read.
[-S FILE]	True if FILE exists and is a socket.

```
if TEST-COMMANDS;
then CONSEQUENT-COMMANDS;
fi
```

./scripts/doccheck.sh

```
#!/bin/bash
echo "This scripts checks the existence of presentation_day1.pdf."
echo "Checking..."
if [ -f ./../doc/presentation_day1.pdf ]
then
echo "The PDF file of this presentation exists."
fi
echo
echo "...done."
echo
```

./scripts/testleapyear.sh

Inside the if statement, you can use another if statement. You may use as many levels of nested ifs as you can logically manage.

```
#!/bin/bash
# This script will test if we're in a leap year or not.
year=`date +%Y`
if [ $[$year % 400] -eq "0" ]; then
echo "This is a leap year. February has 29 days."
elif [ $[$year % 4] -eq 0 ]; then
if [ $[$year % 100] -ne 0 ]; then
echo "This is a leap year, February has 29 days."
else
echo "This is not a leap year. February has 28 days."
fi
else
echo "This is not a leap year. February has 28 days."
fi
```

A script can call an other script or program

```
# call the program feed.sh
# with:


./scripts/feed.sh apple penguin


# check out how they are working
# try an other program call
```

./scripts/WHILE_EXAMPLE1.CSH

```
#!/bin/csh
#
#  WHILE_EXAMPLE1.CSH
#  Use the WHILE command to construct a DO loop.
#
#  To make a DO loop, initialize the counter with a SET command.
#
#  Run the loop with a WHILE command, until the counter is equal to one more
#  than your limit.
#
#  Increment the counter with the @ command.
#
set n = 10
while ( $n != 26 )
  echo $n
  @ n = $n + 1
end
```

./scripts/FOREEACH_EXAMPLE1.CSH

```
#!/bin/csh
#
#  FOREACH_EXAMPLE1.CSH
#  Using the FOREACH command with an explicit list.
#
#  This loop sets I to each value in the list.
#
foreach i ( 10 15 20 40 )
  echo $i
end
#
#  The values don't have to be numeric.
#
foreach i ( a b c 17 )
  echo $i
end
```

UNIX Tutorial for Beginners:
http://www.ee.surrey.ac.uk/Teaching/Unix/index.html

The first UNIX Manual ever:
http://cm.bell-labs.com/cm/cs/who/dmr/1stEdman.html

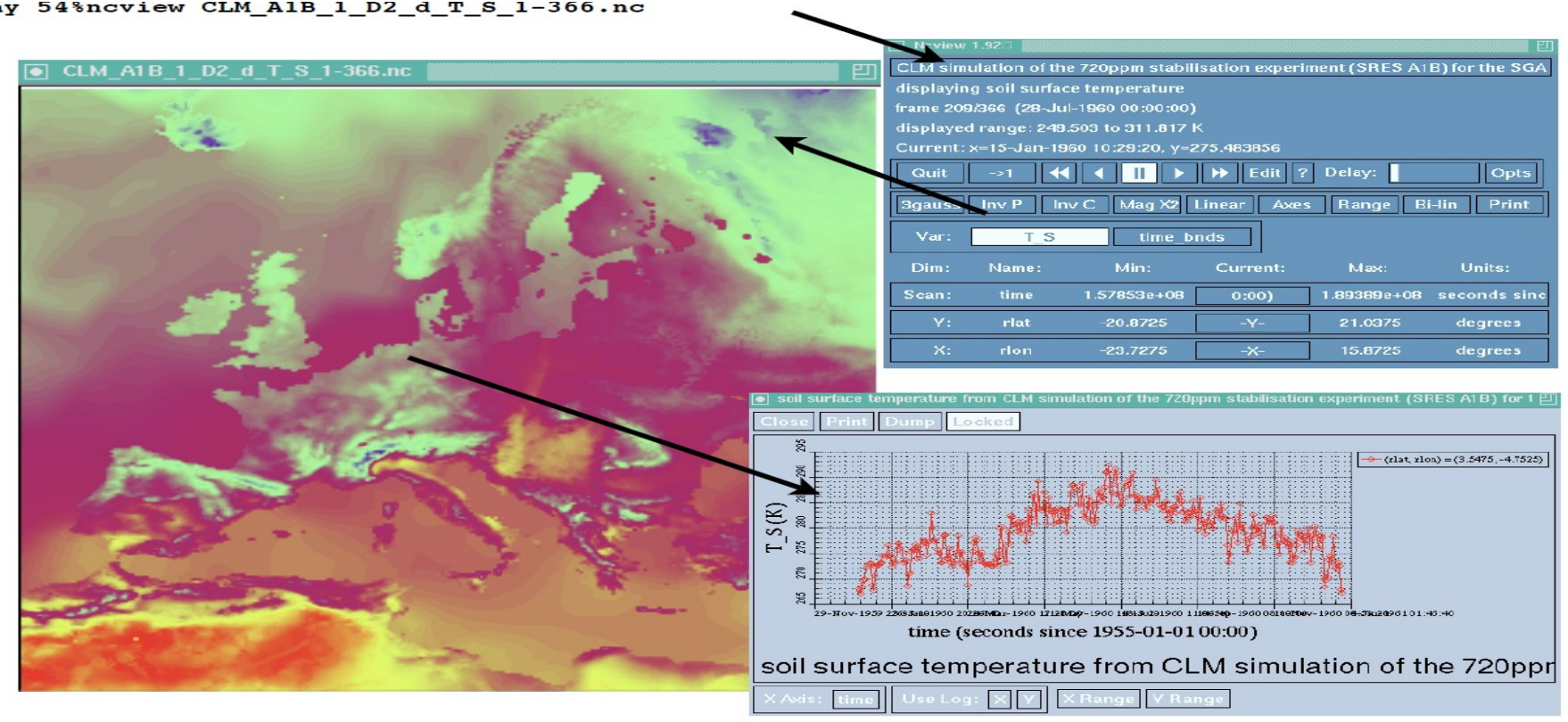
UNIX Guide for Beginners:
http://sillydog.org/unix/

Introduction to UNIX commands:
http://kb.iu.edu/data/afsk.html and much more….

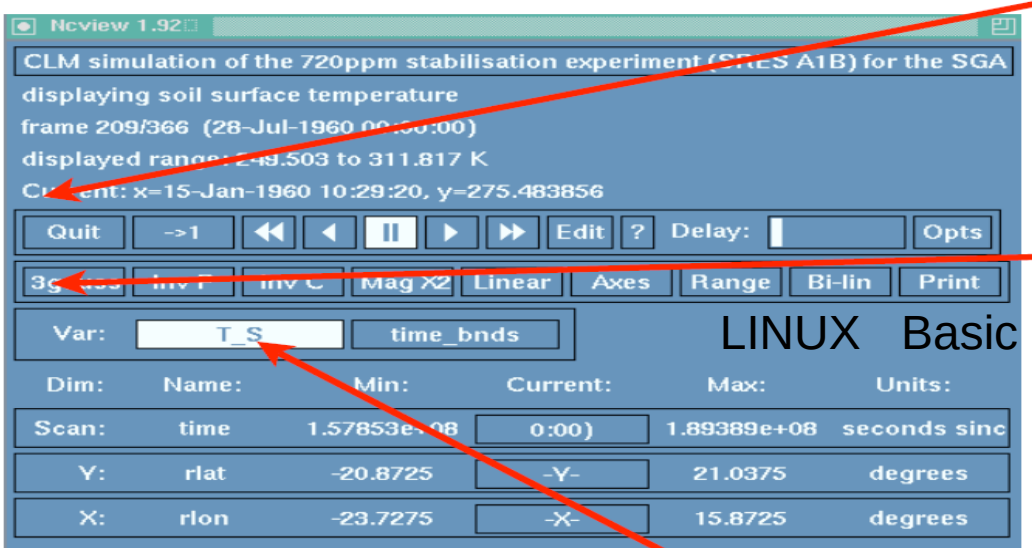
Books:
The Wait Group: UNIX Primer Plus (SAMS)-english
Jerry Peek & all: UNIX Power Tools (O-Reilly)- english

netCDF

```
tiny 54%ncview CLM_A1B_1_D2_d_T_S_1-366.nc
```



ncview



LINUX Basic commands

first row:
1:quit ncview
2:back to the first screen
3 a-e:move through the data
4:edit data
5:infos about the variable
6:delay between two screens
7:contour map overlay!
second row:
08:color tables
09:invert plot (turn upside down)
10:invert colors
11:magnify plot(increase:left, decrease right mouse button click)
12:kind of color filter (hi empha-sizes high, low low values, linear = off)
13:change axes
14:set data range (klick with the right mouse button gives the best range of colors for the plot)
15:bilinear interpolation or replicate pixels
16:print plot to postscriptfile
third row:
choose variable

- **Explore panoply**