Universität Göttingen
Institute for Computer Science / GWDG
Kevin Lüdemann

Exercise Introduction

Before starting the exercise, make sure you have the slide deck for the Linux Crash Course ready and you have a Bash shell under Linux before you. You can use the GWDG machines or any other Linux system with an up to date Bash shell, such as a local virtual machine.

The goal of these exercises is to make you familiar with the Linux system so feel free to play around with it, test things out and either ask for help or search for help online (RTFM first). This crash course focuses on a very small set of use cases. A true test of Linux as an operating system can only be achieved by daily driving it. We recommend that you give it a try for your day to day life, even a VM will give you insight into how you can work with it. These tasks are intended as a primer for the full experience of daily driving Linux.

When copying out commands, depending on the PDF reader you are using, spaces might be lost such that the command does not work. Check with the command in the PDF and add missing spaces.

Contents

Task 1: Follow along the compiling process (20 min)

This exercise walks you through the commands shown in the slides. You do not have to perfectly follow all steps, experiment with the commands if it helps you to get a better understanding of them.
Try using TAB to auto-complete commands and file/directory names.
Use ARROW-UP/DOWN to cycle through your command history and reuse or edit past commands if it means typing less.

`mkdir -p $HOME/apps`	
`mkdir -p $HOME/apps/install`	
`mkdir -p $HOME/apps/install/fftw`	Create the folders for the software
`cd $HOME/apps/install/fftw/`	Switch into the directory
`wget http://www.fftw.org/fftw-3.3.10.tar.gz`	Download the source code
`tar xvzf fftw-3.3.10.tar.gz`	Extract the source code
`module load intel-oneapi-compilers`	Load the compiler (SCC only)
`cd fftw-3.3.10`	Switch into the extracted folder
`./configure CC=icc --prefix=$HOME/apps/fftw-3.3.10`	Configure the install
`make -j 4`	Compile the code on 4 cores

`make check`	Check the compiled code for errors
`make install`	"Install" the code to the defined location
`ls -alh $HOME/apps/fftw-3.3.10/`	Check the installation

Further Reading

- Advanced Programming in the UNIX Environment 3rd edition by R. Stevens and S. Rago

Task 1: For the Advanced (20 min)

This is a more difficult **optional** task which can be done instead of or in addition to Task 1

Work on these tasks once you are done with the first part. You do not need to complete them all or in that order, focus on those that interest you.

- Change the install location to `$HOME/.local/.share/lib/`
- Compile the code with only one core and compare the compile time with the `time` command
- Change the compiler

Task 2: Bash Scripting data collection (30 min)

The idea of this task is to use a bash script for collecting data over time stored in a system file. We will focus on the CPU frequency for now and write it for a single core into a file. This file will contain two columns. The first column is a time stamp like this `20240319_094726` . The second column is the current frequency of core0. The steps of the task are outlined below but you will need to get more knowledge about the commands, so RTFM.

- First you will need to find out about the date command. Run `man date`
- Save the time stamp into a variable
- Find the file that contains the frequency of core0
- Use `cat` to store the frequency in a variable
- Echo the content of the variables into a file with a pipe
- Put all the commands into a bash script
- Write a loop that appends these values every second to the same file

Further Reading

- `https://linuxhint.com/30_bash_script_examples/`

Task 2: Add more cores, free memory and temperatures (30 min)

This is a more difficult **optional** task which can be done instead of or in addition to Task 2

Tracking one core is usually not useful in a many core system. Therefore, you can update the script you have written before and include all cores.

Additionally, the amount of free memory (RAM) is an interesting information to have, especially for debugging memory leaks. Find the file containing the amount of free memory and update the bash script to store this value in an additional column.

Not completely relevant in an HPC setting but for home computers and laptops, use the tool called `sensors` to read out the CPU temperature. If you are running this on a local machine or VM you may need to install the `lm-sensors` package. Use the `grep` command and string manipulation to extract the relevant text. This part is not possible on the SCC since this package is not installed.

Once you are done, think about showing the content of the file graphically. Use a plotting program like `gnuplot` or plotting libraries for python or any other language.