1 Part 1 – Introduction

1.1 A light beginning

Write the name of current working directory.

The UNIX file system forms a hierarchical structure. The “layers” of this structure correspond to directories (folders) containing files and other subdirectories. Directory structure has the form of a *tree* growing up from a so called *root*.

If we want to express a position of a particular file or directory within this tree we have to do that using a *path* which describes how we can get from the root to the file or directory in question. The names of directories on the path are separated using an ordinary slash (“/”).

Each running program has an associated directory — its workind directory. The essence of this exercise is to write down the path to the current working directory of the current shell.

1.2 Adding parameters

Write the contents of directory `/etc` including detailed information on each file.

1.3 Combining for the first time

Write the number of system users.

The list of system users is stored in file `/etc/passwd`. It is a text file with a line corresponding to each user.

1.4 Further redirection

Create a file `datum` (a *date*) containing a line with string “datum”.

1.5 Add subtly

Create a file `datum` consisting of two lines, the first line one contains just a string “Datum:” while the second line contains the current date.

1.6 Take it away again

Remove the first line from the file `datum` created in the previous exercise.

1.7 Cover tracks

Modify the file *datum* from the previous chapter into the original state (i.e. insert the original first line back), but keep the file’s modification date and time.

1.8 A birthday present

Create a file named “darek k narozeninam” (i.e. “a birthday present”) with modification time equal to the date of your birthday.

1.9 An asterisk

Update the modification date and time for all files in the current directory whose names start with “d” and end with “m”.

1.10 Tricky names

Create a file named “-f” and remove it again.

1.11 The first shell program

Count the files in `/etc`.

1.12 Counting files better

Count the files in a directory including the hidden ones.
As “hidden” files we consider those with names starting with a dot.

1.13 Counting subdirectories

Count the subdirectories of a directory (not regular files).

1.14 Counting in depth

Count all files in the whole subtree rooted at the current directory.

1.15 A range of lines

Write the lines 6 to 10 from the file `/etc/passwd`.

1.16 Columns

Write the list of users in the following form: “login=NAME SURNAME”

1.17 A line from a column

Write the list of names of user groups into one line with names separated with a space.
The list of groups is contained in the file “/etc/group” which is again a text file.
The first field contains a group name, the third a group id (GID), and the fourth contains a list of group members.

1.18 And vice versa

Write the list of names of groups you belong to into a column.

1.19 Cutting and pasting differently

Modify the output of command `id` so that the field “`uid=`” is moved to the end.

1.20 The list of groups without a help

Write the list of group ids to which you belong, but without using the `id` command.

1.21 For what scissors are not enough

Write the list of files in the current directory in the following form: “size name”.

Further exercises

1. Write the list of files in directory `/etc` which have a suffix. After that write the same list but now filter file names from the output of "**`ls /etc`**".
2. Create two files named `one` and `two`. Write a command which outputs the name of the older one (and does not output the name of the younger one as well).
3. The file `.bash_history` in your home directory (if you are using other shell than **`bash`** find the correct name of a history file in a corresponding man page) contains several last commands you have used (they are being stored there at the time of logout from the system). List all commands stored in this history in which you have used a pipe.
4. In Chapter 3.16 we shall speak about RFC documents and `rfc-index.txt` file. Find in this file all lines containing string "-H".
5. Find out the number of users in the system with the same name as your.
6. Command **`ps`** outputs information on running programs. Try it out and choose a name of a running program. Try to write a command which filters out the lines with the chosen program from the output of **`ps`**. Run it several times and you may discover that the unwanted line with **`grep`** command used to filter the lines appears as the part of the filtered output as well. Find out why this line occurs and modify your filtering command so that this does not happen.
7. Write the current time. You can use **`date`** command but only in the form with no parameters.
8. Write the number of members of the `wheel` group stored in `/etc/group`.
9. Solve the exercise from Chapter 1.19 by using **`cut`** and **`paste`** only.
10. In Chapter 1.20 we have (for the sake of clarity) not consider the case when the string we were looking for appears in other fields of `/etc/passwd` and `/etc/group` lines, than we need. Modify the program so that the user name is really searched for only in the first, resp. fourth column.
11. Solve the exercises in Chapters 3.4 and 3.8 with tools we have gone over until now.
12. Try to improve overall impression of the solution of exercise in Chapter 1.17 by first removing the comment lines starting with "`#`" from the `/etc/group` file.
13. Command **`who`** outputs the users who are currently logged in together with further information on these users. List the user names of these users in one line.

2 Part 2 – Working with files

2.1 A mailbox

Create a directory to which anyone can add a new file but nobody except you can determine the names of files from other users.

2.2 A new command for the sclerotic ones

Copy file `/bin/mv` to your directory and rename it to `rename`. Change its permissions to the least possible subset which allows you to run it as a command.

2.3 A new command for the lazy ones

Create a new shell script named `ll` which will call “**`ls -ltr`**”.

2.4 A new command for the dubious ones

Add a new name “`ltr`” to script `ll` from the previous exercise.

2.5 Copying links

Copy files `ll`, `ltr.h` (hard link to `ll` created in the previous exercise), and `ltr.s` (soft link to `ll` created in the previous exercise) to directory `bin`.

2.6 Moving links

Move files `ltr.h` and `ltr.s` to a new directory named `linky`.

2.7 How to create nothing

Create or modify file `nic` (=`nothing`) so that it is empty (its length is 0). Your command should work regardless on whether file `nic` has already existed or not.

2.8 A command for Dr. Watson

In Chapter 2.2 we have made a copy of **`mv`**. We assumed that we know its solution. Modify your command so that it works even without this knowledge.

2.9 Not this way, dear Watson

Find all directories named `lib` which are on the “second level” of the file hierarchy (i.e. in some directory which is in the root directory) and output a “long” information on each of these file using command **`ls`**.

2.10 To render to every man his due

Assign the (minimum possible) permitions to all files and directories in a particular subtree so that anyone who is not the owner (others) can read them. Use e.g. a subtree rooted in your home directory provided you do not have ther any sensitive data.

2.11 Who is who

List all shell scripts in the subtree rooted at /etc.

To recognize a shell script use command **file** which can output to each a character of its contents, e.g:

```
sinek:~> file /etc/rc
/etc/rc: Bourne shell script text executable
```

2.12 Sorting

Output a list of user ids (UID) sorted in an increasing order.

2.13 Sorting with a key

Output a list of user names sorted by their UIDs in an increasing order.

2.14 Sorting with problems

Output the lines of /etc/passwd sorted by a surnames and given names of users.

Suppose the standard form "…:Name Surname:…" of the fifth column and ignore possible deviations to this format (such as multiple given names, spaces in other columns etc.)

2.15 Sort "somewhere else"

Output a subset of lines in /etc/passwd containing the five users with the highest UID.

The lines have to be output in the order in which they occur original file.

2.16 Unique sort

List the users in the group with number zero.

2.17 Find ten differences

List the duplicit UIDs in the system.

2.18 At the visit to databases

List the system users, add a numbers of their primary groups to each name.

2.19 Input vs. parameters

Delete the youngest file in a directory.

You can create it first (nejmladsi_soubor means youngest_file):

```
sinek:~> touch nejmladsi_soubor
```

Further exercises

1. Modify the programs solving exercises in chapters 2.12 and 2.13 so that the comment lines in /etc/passwd starting with character "#" are ignored.

2. Try to sort the output of **`ls`** command in directory `/bin` according to file size.
   Hint: Look in Chapter 2.13 for an inspiration.
   And try to really program the solution, study the functionality of "**`-S`**" option of **`ls`** later…

3. At the very end of Chapter 3.4 we shall return to the exercise from Chapter 2.14 and we shall solve it using an external sort key appended as the first column of a file. You can do it even now without using an editor.

4. Solve the exercise from Chapter 3.22 using the tools we know at this time.
   Hint: Here also an external sort key can help.

5. Find all files in directory `/etc` with suffix "`conf`" and find out which of them has the biggest number of lines.

6. Choose a not too big subtree of the file system (e.g. a subtree under your home directory) and find a biggest file in it. If you find several possible solution, compare their speed.

7. We have seen the **`who`** within *Further exercises* to Part 1. It outputs a list of users who are currently logged in. If a user is logged in multiple times, he/she is present multiple times in the list. Find a user which is currently logged in the biggest number of times.
   Hint: The key role should be played by command **`uniq`**.

8. Create two directories `one` and `two`, and copy the same group of files to both of them. Then some of the files delete in both of these directories (different files in each directory). Write a script which copies the deleted files from each directory to the other one.

9. Home directories of all users are usually concentrated into one parent directory (often called `/home`). Output a list of files (subdirectories) in this directory and to each directory write the UID of its owner.
   Hint: Think how the **`join`** command could be used to your advantage.
   Solution using "**`-n`**" option of command **`ls`** is not what we have on mind.

10. Solve the exercise from Chapter **??** with help of command **`xargs`**.

3 Part 3 – Editors

3.1 Déjà vu

Output the lines of `/etc/passwd` sorted according to surnames and given names of users.

Suppose the standard form "…:Name Surname:…" of the fifth column and ignore possible deviations to this format (such as multiple given names, spaces in other columns etc.)

3.2 Line numbers

Select (with a single command) the fifth line of file `/etc/passwd`.

3.3 A more complex selection

Output the list of users with UID between 1000 and 1199.

3.4 Columns reordering

Output the list of users "*Surname Name*".

3.5 Finding duplicities

List those lines of `/etc/passwd` which contain a UID consisting of at most three digits and being equal to GID.

3.6 Finding numbers

Select all substring representing a correct IP address from the standard input.

An IP address (an address of computers within a TCP/IP network) consists of four numbers (between 0 and 255) separated with dots.

3.7 Finding words

Find the groups having `root` as a member in `/etc/passwd`.

3.8 Shooting at a moving target

Find those groups in `/etc/group` which have users `root` or `forst` as members.

The list of groups should be output in a form of "*group_name:GID:members_list*".

3.9 Next, please

Output a column contaings numbers of groups you are a member of.

Start with the output of the **id** command as it was broken apart in Chapter 1.19.

```
sinek:~> id | tr ' =' '[\n*]'
…
groups
1004(forst),0(wheel),1(daemon),999(kernun),1028(j28)
```

3.10 A cycle

Write a program which expects strings containing (absolute and correct) paths within file system at its standard input and "normalizes" these paths.

An example of the modification:

```
    /etc/././passwd            ⇒       /etc/passwd
    /usr/local/lib/../src      ⇒       /usr/local/src/
```

3.11 Adding lines

List the system administrators (users with UID equal to 0).

For each administrator output two lines in the following form:

```
Administrator:                        …fixed text
root:*:0:0:Super User:/root:/bin/bash   …line of /etc/passwd
```

3.12 Left-luggage office

Output the list of system administrators (users with UID equal to 0) in the following form:

```
Administrator root:                        ...text with user name
root:*:0:0:Super User:/root:/bin/bash      ...line of /etc/passwd
```

3.13 What do they resent at a train station

Split given simple text at the input to paragraphs using HTML tags "`<p>`" and "`</p>`". The paragraphs in the input text are separated with one or more empty lines.

HTML is a language in which web pages are being written. It is a *markup* language, its syntax is formed with different tags using which an author of a web page specifies her/his wishes on how the contents of the web page should be formatted (for more see in [9]). Tag "`p`" (*paragraph*) a request to create a paragraph is specified. A text of a paragraph is enclosed between a pair of tags "`<p>`" and "`</p>`".

This exercise comes from book [3].

3.14 Once more into a left-luggage office

Output a list of users in a form of "login_name=NAME SURNAME".

3.15 Wrapped lines

Write a program which in the input text modifies "`Copyright 2000-2008`" to "`Copyright 2000-2009`". The substitution should work even in the case when word "`Copyright`" is at the end of a line.

3.16 Memory

Write a program which outpus the list of invalidated RFC documents.

RFC (Request for Comments) is a set of documents describing and defining behaviour of the Internet network. They can be found on most internet servers offering publicly available documents, e.g. at `ftp://sunsite.mff.cuni.cz/Network/RFCs`.

As the time goes older documents become naturally invalid. The purpose of this exercise is to find in the indx file (`rfc-index.txt`, download it) numbers of those RFCs, which are marked as "Obsoleted". The format of an index file is as follows:

```
0172 The File Transfer Protocol. A. Bhushan, B. Braden,
     W. Crowther, E. Harslem, J. Heafner, A. McKenzie,
     J. Melvin, B. Sundberg, D. Watson, J. White. June 1971.
     (Format: TXT=21328 bytes) (Obsoleted by RFC0265)
     (Updates RFC0114) (Updated by RFC0238)
     (Status: UNKNOWN)

0173 ...
```

Each RFC has one corresponding record (block of lines) ended with an empty line. The first line contains a number, possible "`Obsoleted`" attribute can be found on one of the subsequent lines.

3.17 Vertical mirror

Write a program which reverses the contents of each input line.

At some systems a command with this functionality is present under the name **`rev`**.

3.18 A folding picture-book

Write a program which splits an arithmetical expression given at the only input line so that individual atoms (natural numbers, brackets and operators) are at separate lines.

3.19 File editing

Assume that you have a file `datum` containing the current date as it was created in Chapter 1.6. Or you can create such file ("**`date > datum`**").

Using an editor insert a line "`Datum:`" at the beginning of the file.

3.20 Current line

Store the output of command **`id`** split into lines as in Chapter 1.19 to a file named `id.out`:

```
sinek: > id | tr ' =' '[\n*]' > id.out
sinek: > cat id.out
…
gid
1004(forst)
…
```

Modify this file using an editor so that the line containing information on primary group are deleted (the line immediately following the line with "`gid`").

3.21 Globalization

Delete the records of invalidated RFC documents from file `rfc-index.txt` (see Chapter 3.16).

3.22 Horizontal mirror

Output the lines of `/etc/passwd` in the reversed order (the first line as the last one).

At some systems such command exists under name **`tac`**.

3.23 A Machine

Write an **`ed`** script which modifies the normal form output of **`diff`** to the form which would be output with option "-e".

We have already seen command **`diff`**, it compares two files and outputs the lines in which the two files differ. It has several modes of output. The normal form of output looks as follows:

```
sinek:~> cat old
o-1
o-2
o-3
o-4
o-5
o-6
sinek:~> cat new
o-3
n-x
n-y
o-6
o-z
sinek:~> diff old new

1,2d0    ...l. 1 to 2 from file old are missing in file new, they would go after l. 0
< o-1        ...line No. 1 from file old
< o-2        ...line No. 2 from file old
4,5c2,3  ...l. 4 to 5 from old differ from the corresponding l. 2 to 3 from new
< o-4        ...line No. 4 from file old
< o-5        ...line No. 5 from file old
—        a separating line
> n-x        ...line No. 2 from file new
> n-y        ...line No. 3 from file new
6a5      ...l. 5 from file new is missing in old, it would go after l. 6
> n-z        ...line No. 5 from file new
```

If **`diff`** is called with option "**`-e`**", it lists a series of commands for editor **`ed`** – if this series is applied to the first file, the result would be the second file. It is kind of a machine which can be repeatedly used to convert one file to another.

In our case the output would look as follows:

```
sinek:~> diff -e old new
6a
n-z
. 4,5c
n-x
n-y
.
1,2d
```

Further exercises

To be added.