IST 615 – Cloud Management

School of Information Studies, Syracuse University

Fall 2024

Homework - 1

Name of the Author: **Karan Shah**

Assignment Due: 9/10/2024

Assignment Submitted: 9/8/2024

(The document has **4 pages** with the cover page)

Question-1

Submit a screenshot of the results of three commands that you executed as you were going through the course and clearly explain the command that you used and its output.

1. **Awk** is a potent command-line utility for handling and examining text files. It is especially helpful for reformatting or converting data according to rules and for extracting particular data from structured files, such logs or CSV files.
   For Example we have a text file simple_data.txt which has below table

Name	**ID**	**Team**
Scott	314	Purple
Ananti	991	Orange
Jian	3127	Purple
Miguel	671	Green
Wes	1337	Orange
Anne	556	Green

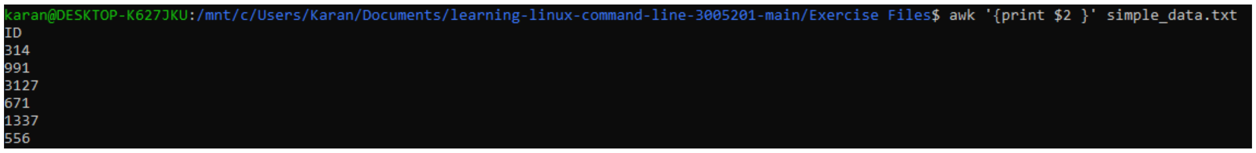
To extract the second column, which contains the IDs, we can use the following `awk` command:

Awk – command that processes the file

'{print $2}' – Instruction that tells awk to print second column of line in the file

simple_data.txt – file that awk is processing

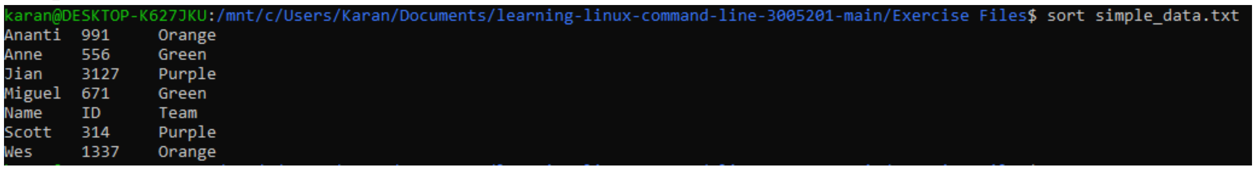
Output:



```
karan@DESKTOP-K627JKU:/mnt/c/Users/Karan/Documents/learning-linux-command-line-3005201-main/Exercise Files$ awk '{print $2 }' simple_data.txt
ID
314
991
3127
671
1337
556
```

2. **Sort** command in Linux is used to arrange lines of text files in a specified order. By default, it sorts text lines alphabetically. Consider the same table as above, the sort command will have the names sorted in alphabetical order as shown below:

Output:

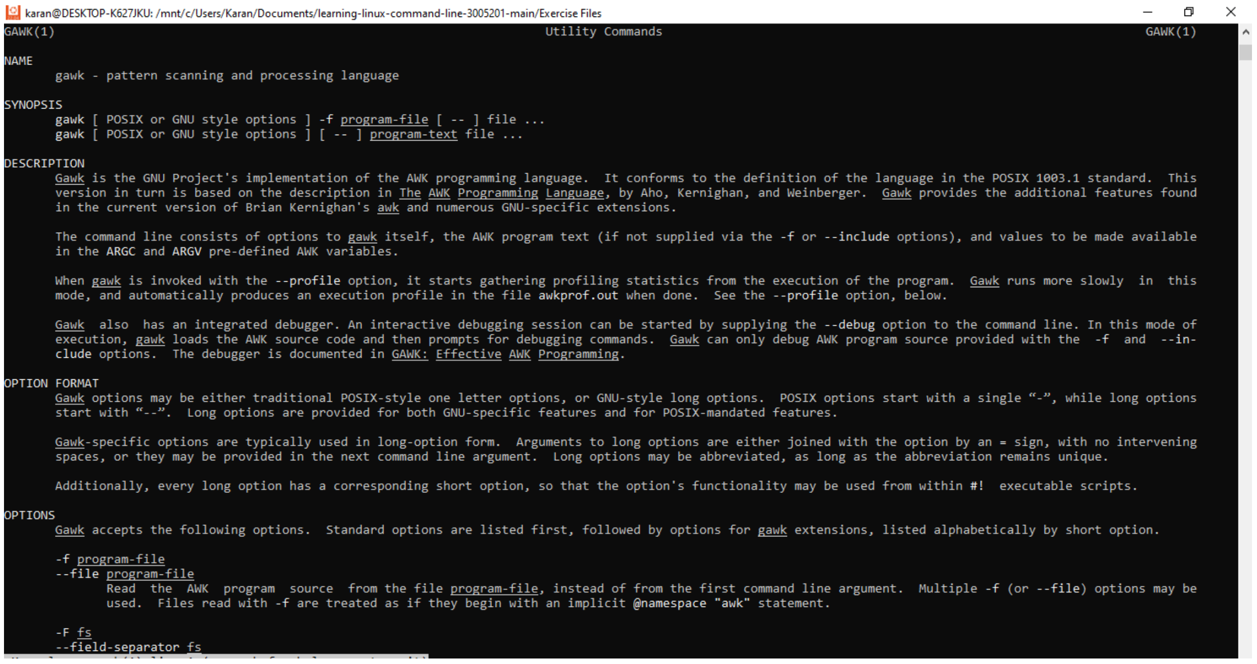


```
karan@DESKTOP-K627JKU:/mnt/c/Users/Karan/Documents/learning-linux-command-line-3005201-main/Exercise Files$ sort simple_data.txt
Ananti  991     Orange
Anne    556     Green
Jian    3127    Purple
Miguel  671     Green
Name    ID      Team
Scott   314     Purple
Wes     1337    Orange
```

3. **Man** command helps us referring the documentation. This was one of the most used commands by me in the course. For instance to understanding the difference between awk and sed, I used command Man

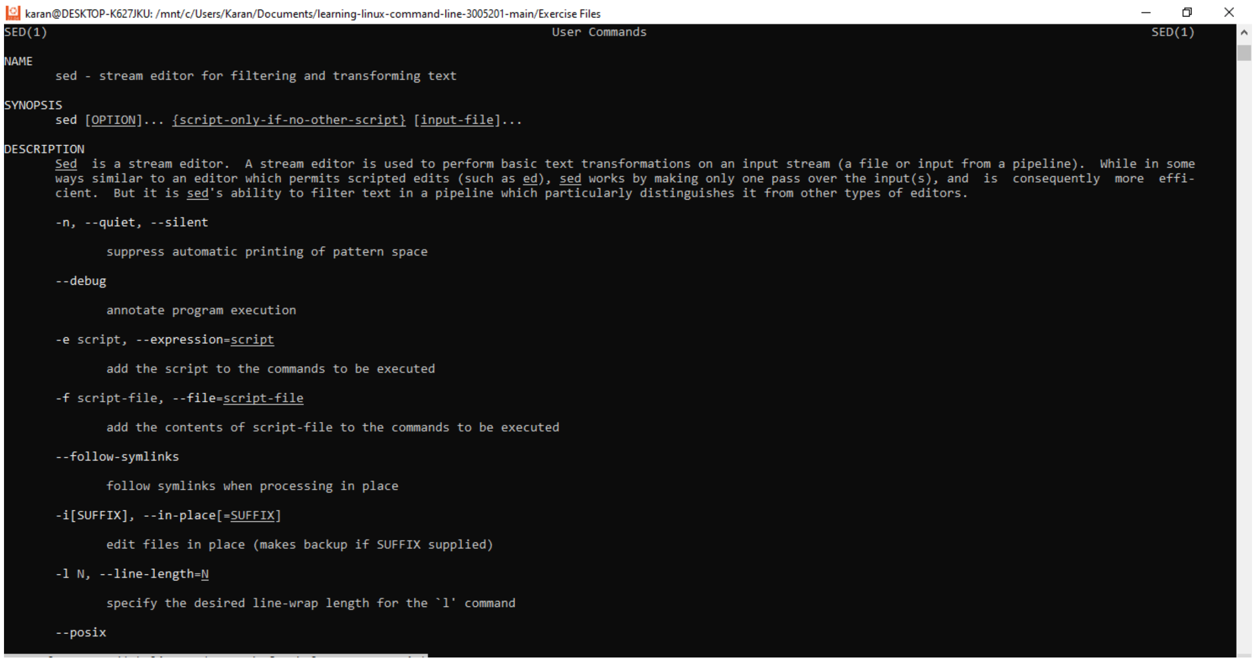
Output

Man awk



```
karan@DESKTOP-K627JKU: /mnt/c/Users/Karan/Documents/learning-linux-command-line-3005201-main/Exercise Files
GAWK(1)                                        Utility Commands                                        GAWK(1)

NAME
       gawk - pattern scanning and processing language

SYNOPSIS
       gawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...
       gawk [ POSIX or GNU style options ] [ -- ] program-text file ...

DESCRIPTION
       Gawk is the GNU Project's implementation of the AWK programming language.  It conforms to the definition of the language in the POSIX 1003.1 standard.  This
       version in turn is based on the description in The AWK Programming Language, by Aho, Kernighan, and Weinberger.  Gawk provides the additional features found
       in the current version of Brian Kernighan's awk and numerous GNU-specific extensions.

       The command line consists of options to gawk itself, the AWK program text (if not supplied via the -f or --include options), and values to be made available
       in the ARGC and ARGV pre-defined AWK variables.

       When gawk is invoked with the --profile option, it starts gathering profiling statistics from the execution of the program.  Gawk runs more slowly  in  this
       mode, and automatically produces an execution profile in the file awkprof.out when done.  See the --profile option, below.

       Gawk  also  has an integrated debugger. An interactive debugging session can be started by supplying the --debug option to the command line. In this mode of
       execution, gawk loads the AWK source code and then prompts for debugging commands.  Gawk can only debug AWK program source provided with the  -f  and  --in‐
       clude options.  The debugger is documented in GAWK: Effective AWK Programming.

OPTION FORMAT
       Gawk options may be either traditional POSIX-style one letter options, or GNU-style long options.  POSIX options start with a single “-”, while long options
       start with “--”.  Long options are provided for both GNU-specific features and for POSIX-mandated features.

       Gawk-specific options are typically used in long-option form.  Arguments to long options are either joined with the option by an = sign, with no intervening
       spaces, or they may be provided in the next command line argument.  Long options may be abbreviated, as long as the abbreviation remains unique.

       Additionally, every long option has a corresponding short option, so that the option's functionality may be used from within #!  executable scripts.

OPTIONS
       Gawk accepts the following options.  Standard options are listed first, followed by options for gawk extensions, listed alphabetically by short option.

       -f program-file
       --file program-file
              Read  the  AWK  program  source  from the file program-file, instead of from the first command line argument.  Multiple -f (or --file) options may be
              used.  Files read with -f are treated as if they begin with an implicit @namespace "awk" statement.

       -F fs
       --field-separator fs
```

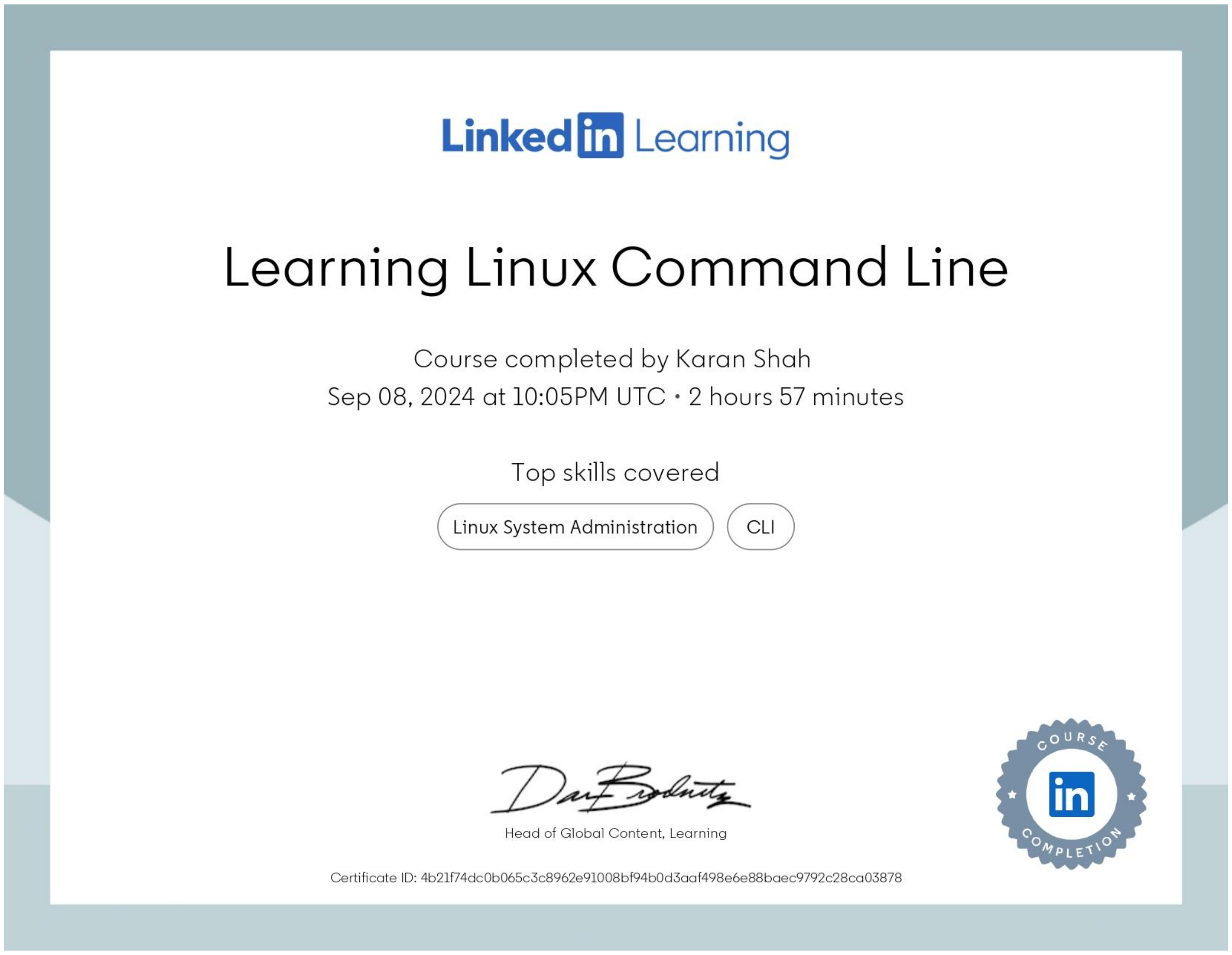
Man sed



```
karan@DESKTOP-K627JKU: /mnt/c/Users/Karan/Documents/learning-linux-command-line-3005201-main/Exercise Files
SED(1)                                          User Commands                                          SED(1)

NAME
       sed - stream editor for filtering and transforming text

SYNOPSIS
       sed [OPTION]... {script-only-if-no-other-script} [input-file]...

DESCRIPTION
       Sed  is a stream editor.  A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline).  While in some
       ways similar to an editor which permits scripted edits (such as ed), sed works by making only one pass over the input(s), and  is  consequently  more  effi‐
       cient.  But it is sed's ability to filter text in a pipeline which particularly distinguishes it from other types of editors.

       -n, --quiet, --silent

              suppress automatic printing of pattern space

       --debug

              annotate program execution

       -e script, --expression=script

              add the script to the commands to be executed

       -f script-file, --file=script-file

              add the contents of script-file to the commands to be executed

       --follow-symlinks

              follow symlinks when processing in place

       -i[SUFFIX], --in-place[=SUFFIX]

              edit files in place (makes backup if SUFFIX supplied)

       -l N, --line-length=N

              specify the desired line-wrap length for the `l' command

       --posix
```

Question-2

Submit the course completion certificate that you will receive at the end of the course (or a screenshot of it)



Linked in Learning

Learning Linux Command Line

Course completed by Karan Shah

Sep 08, 2024 at 10:05PM UTC • 2 hours 57 minutes

Top skills covered

Linux System Administration

CLI

[illegible]

Head of Global Content, Learning

Certificate ID: 4b21f74dc0b065c3c8962e91008bf94b0d3aaf498e6e88baec9792c28ca03878

COURSE COMPLETION