# BINF2111 – Introduction to Bioinformatics Computing

## UNIX 101 – enter the coding zone


**Richard Allen White III, PhD**
**RAW Lab**

**Lecture 5 - Tuesday Sep 6th, 2022**

# Learning Objectives

- Review bonus

- Comments about data and formats

- Sort/uniq/cut commands

- PATHS

- Quiz 5

# Bonus 4

- Delete all the empty lines in the empty lines file with

- Delete all the ‘all white space’ with grep

Also, in python (think Pandas)

# Bonus 4

- Delete all the empty lines in the empty lines file with
→ grep: grep -v -e '^$' file

- Delete all the ‘all white space’ with grep

Also, in python (think Pandas)

# Bonus 4

- Delete all the empty lines in the empty lines file with
→ grep: grep -v -e '^$' file
→ awk:  awk '!/^$/' file

- Delete all the ‘all white space’ with grep

Also, in python (think Pandas)

# Bonus 4

- Delete all the empty lines in the empty lines file with
→ grep: grep -v -e '^$' file
→ awk:  awk '!/^$/' file

- Delete all the ‘all white space’ with grep
→ grep: grep -v -e '^[[:space:]]*$' file

Also, in python (think Pandas)

# Bonus 4

- Delete all the empty lines in the empty lines file with
→ grep: grep -v -e '^$' file
→ awk:  awk '!/^$/' file

- Delete all the ‘all white space’ with grep
→ grep: grep -v -e '^[[:space:]]*$' file
→ awk: awk 'NF > 0' file
→ sed 's/\t/,/g' empty_lines.txt >empty_lines.csv

Also, in python (think Pandas)
**Any one try in Python Pandas.**

# Bonus 4 – Python Pandas

- First use sed to convert from tsv (tab delim) to csv (column delim)

```
import pandas as pd

df = pd.read_csv('data.csv')
new_df = df.dropna()  or df.dropna(inplace = True)
print(new_df.to_string()) or print(df.to_string())
```

# General comments - data

**UNDERSTAND YOUR DATA**

Expectations - Format, Types, Domains, Uniqueness, etc.
Assumptions - Does it make sense to use data from X in a new context Y?
Restrictions - Licenses, Embargos, etc.

**STANDARDIZE YOUR DATA**

Values - ID mapping, Unit conversion, Adding/Removing Prefixes/Suffixes, etc.
Columns - Adding, Removing, Rearranging, Merging, Splitting, etc.
Rows - Filtering data (duplicates, missing data, not relevant to task, etc)

**RECORD YOUR ANALYSIS PROCESS**

Your computer is a lab - Keep a lab notebook!
Use Version Control like GitHub or BitBucket.

# General comments - data formats

**Column-oriented data**

Spreadsheets, CSV, Tabular, delimited

Fixed position

One line per record, fixed set of fields (columns)

**Key-Value data**

Multiple lines per record

Variable number of fields/values

**Hierarchical data (XML, JSON, Ontologies, etc)**

Nested - usually requires a more complex parser

Usually follow a well-defined schema.

**Web-based API resources**

Issue specific commands to get different types of data

**Infinite proprietary formats**

Usually harder to parse but generally act like other types

# General comments: Column-Oriented

**Easy to browse/explore**
- MOST files are this type

**Best for databases**
- Delimiters are tabs or commas between fields
- 1 line = 1 record
- Each record has fixed set of fields

**Complex code not needed:**
- Excel can handle small datasets (<1 million rows)
- UNIX can handle the rest

**CSV**

david,abdul,xi,bill
mary,david,bill,abdul
wang,abdul,xi,david

**TSV**

David abdul xi bill
Mary david bill abdul
Wang abdul xi david

# General comments - XML formats

**Machine-readable**
- Many tools and libraries available to parse it

**Can represent complex structures**
- multiple values
- nested structures

**Not trivial for non-coders**

```
<?xml version="1.0"?>
<!DOCTYPE Entrezgene-Set PUBLIC "-//NLM//DTD NCBI-Entrezgene, 21st Ja
<Entrezgene-Set>

<Entrezgene>
  <Entrezgene_track-info>
    <Gene-track>
      <Gene-track_geneid>4336</Gene-track_geneid>
      <Gene-track_status value="live">0</Gene-track_status>
      <Gene-track_create-date>
        <Date>
          <Date_std>
            <Date-std>
              <Date-std_year>1998</Date-std_year>
              <Date-std_month>8</Date-std_month>
              <Date-std_day>27</Date-std_day>
            </Date-std>
          </Date_std>
        </Date>
      </Gene-track_create-date>
      <Gene-track_update-date>
        <Date>
          <Date_std>
            <Date-std>
              <Date-std_year>2016</Date-std_year>
              <Date-std_month>12</Date-std_month>
              <Date-std_day>6</Date-std_day>
            </Date-std>
          </Date_std>
        </Date>
      </Gene-track_update-date>
    </Gene-track>
  </Entrezgene_track-info>
  <Entrezgene_type value="protein-coding">6</Entrezgene_type>
  <Entrezgene_source>
    <BioSource>
      <BioSource_genome value="genomic">1</BioSource_genome>
      <BioSource_origin value="natural">1</BioSource_origin>
      <BioSource_org>
        <Org-ref>
          <Org-ref_taxname>Homo sapiens</Org-ref_taxname>
          <Org-ref_common>human</Org-ref_common>
          <Org-ref_db>
            <Dbtag>
              <Dbtag_db>taxon</Dbtag_db>
              <Dbtag_tag>
                <Object-id>
                  <Object-id_id>9606</Object-id_id>
                </Object-id>
              </Dbtag_tag>
            </Dbtag>
          </Org-ref_db>
          <Org-ref_syn>
            <Org-ref_syn_E>humans</Org-ref_syn_E>
            <Org-ref_syn_E>man</Org-ref_syn_E>
          </Org-ref_syn>
          <Org-ref_orgname>
            <OrgName>
              <OrgName_name>
```

# General comments: Key-value formats

**Easy to read**

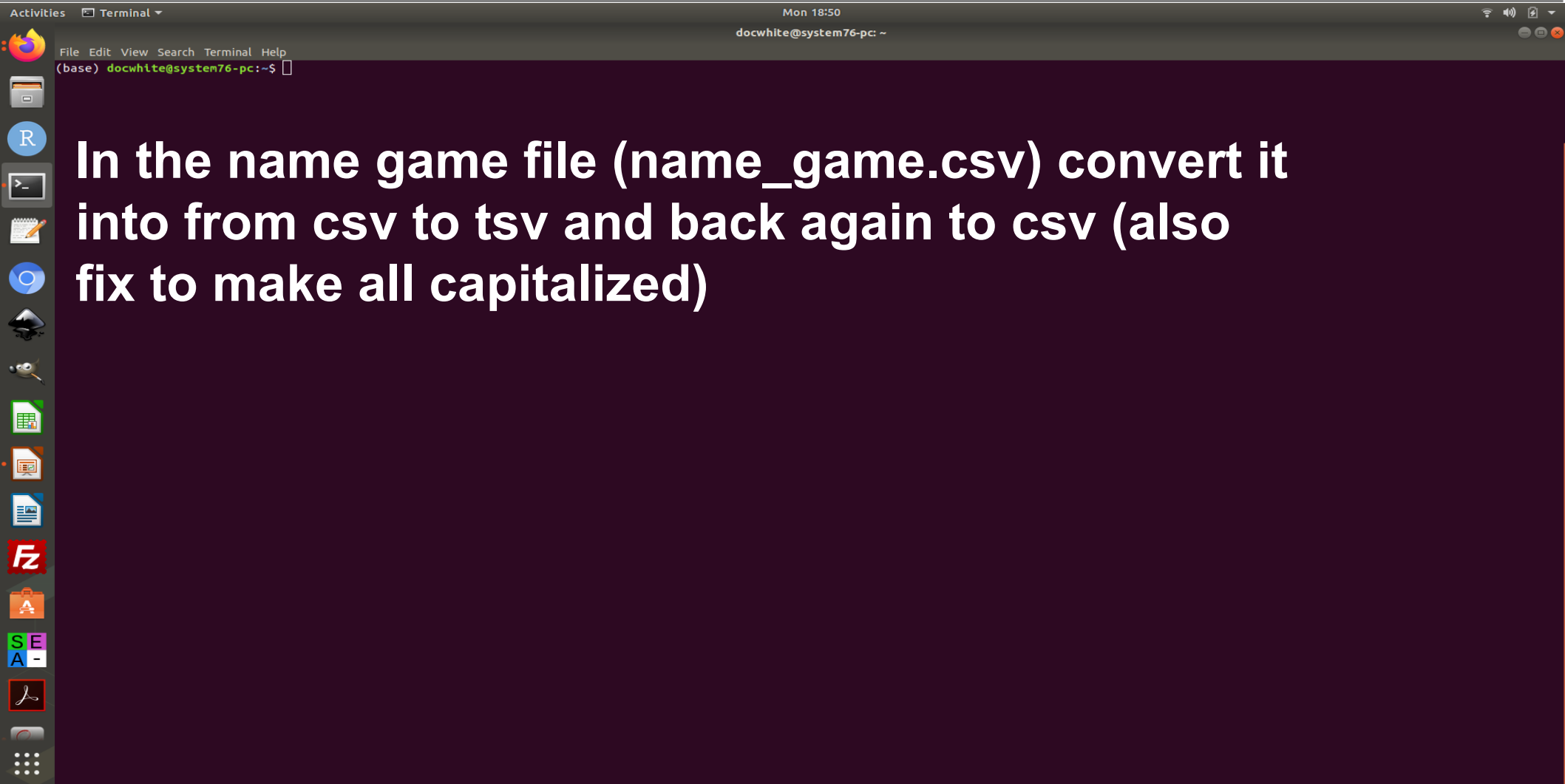
**Straightforward to parse**

- Delimiter for each Record
- One key-value pair per line
- Supports multiple values

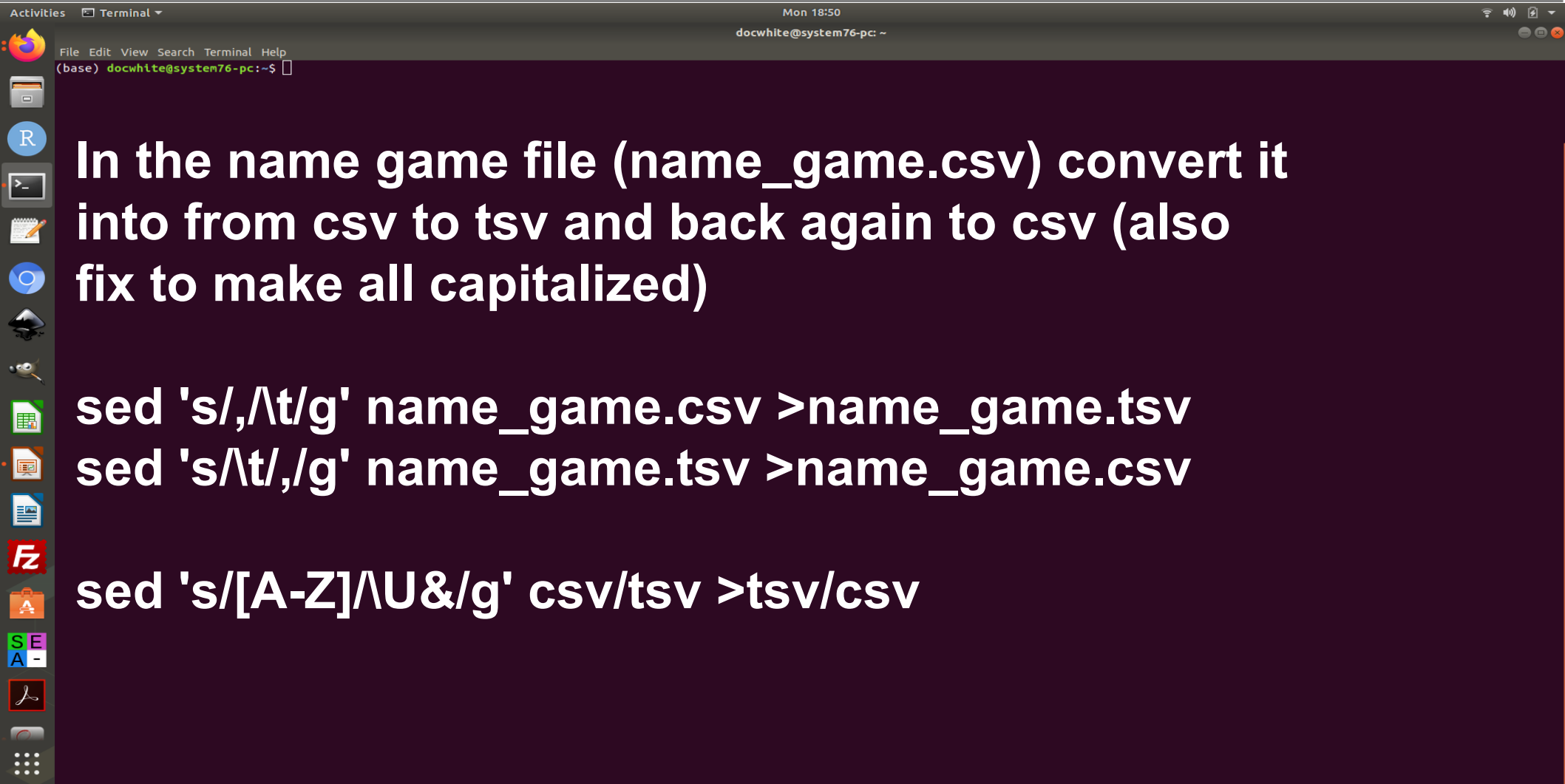
**Novice coders can handle it!**

```
remark: cvs version: use data-version
ontology: go

[Term]
id: GO:0000001
name: mitochondrion inheritance
namespace: biological_process
def: "The distribution of mitochondria, including the mitocho
PMID:10873824, PMID:11389764]
exact_synonym: "mitochondrial inheritance" []
is_a: GO:0048308 ! organelle inheritance
is_a: GO:0048311 ! mitochondrion distribution

[Term]
id: GO:0000002
name: mitochondrial genome maintenance
namespace: biological_process
def: "The maintenance of the structure and integrity of the m
is_a: GO:0007005 ! mitochondrion organization

[Term]
id: GO:0000003
name: reproduction
namespace: biological_process
alt_id: GO:0019952
alt_id: GO:0050876
def: "The production of new individuals that contain some por
subset: goslim_chembl
subset: goslim_generic
subset: goslim_pir
subset: goslim_plant
subset: gosubset_prok
exact_synonym: "reproductive physiological process" []
xref_analog: Wikipedia:Reproduction
is_a: GO:0008150 ! biological_process

[Term]
id: GO:0000005
name: obsolete ribosomal chaperone activity
namespace: molecular_function
def: "OBSOLETE. Assists in the correct assembly of ribosomes
PMID:12150913]
comment: This term was made obsolete because it refers to a c
exact_synonym: "ribosomal chaperone activity" []
is_obsolete: true
consider: GO:0042254
consider: GO:0044183
consider: GO:0051082
```

# Column-oriented fixes examples



# Column-oriented fixes examples




In the name game file (name_game.csv) convert it into from csv to tsv and back again to csv (also fix to make all capitalized)

```
sed 's/,/\t/g' name_game.csv >name_game.tsv
sed 's/\t/,/g' name_game.tsv >name_game.csv
```

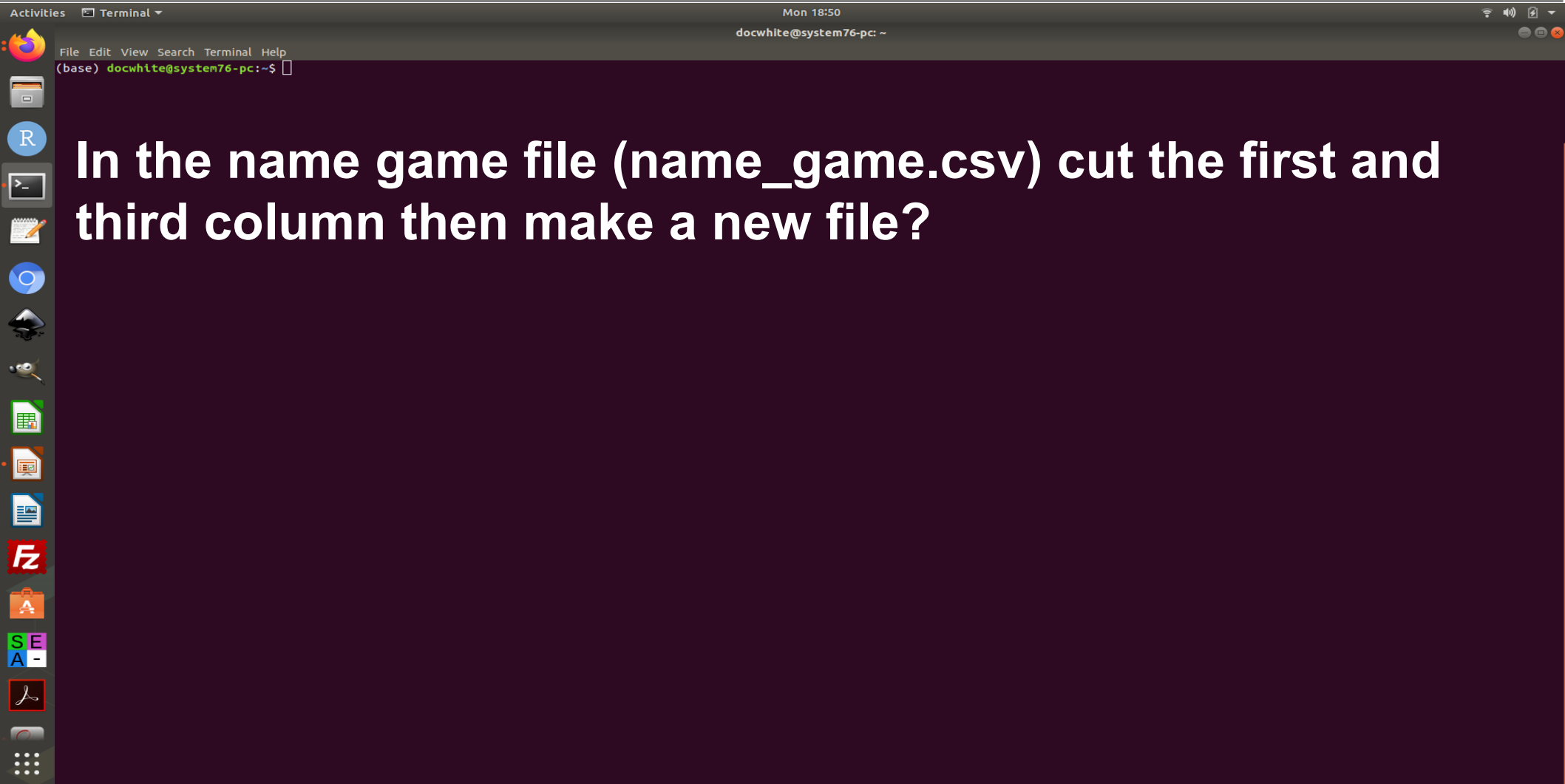
```
sed 's/[A-Z]/\U&/g' csv/tsv >tsv/csv
```

# cut – syntax anatomy UNIX's scissors

# cut [options] file.txt

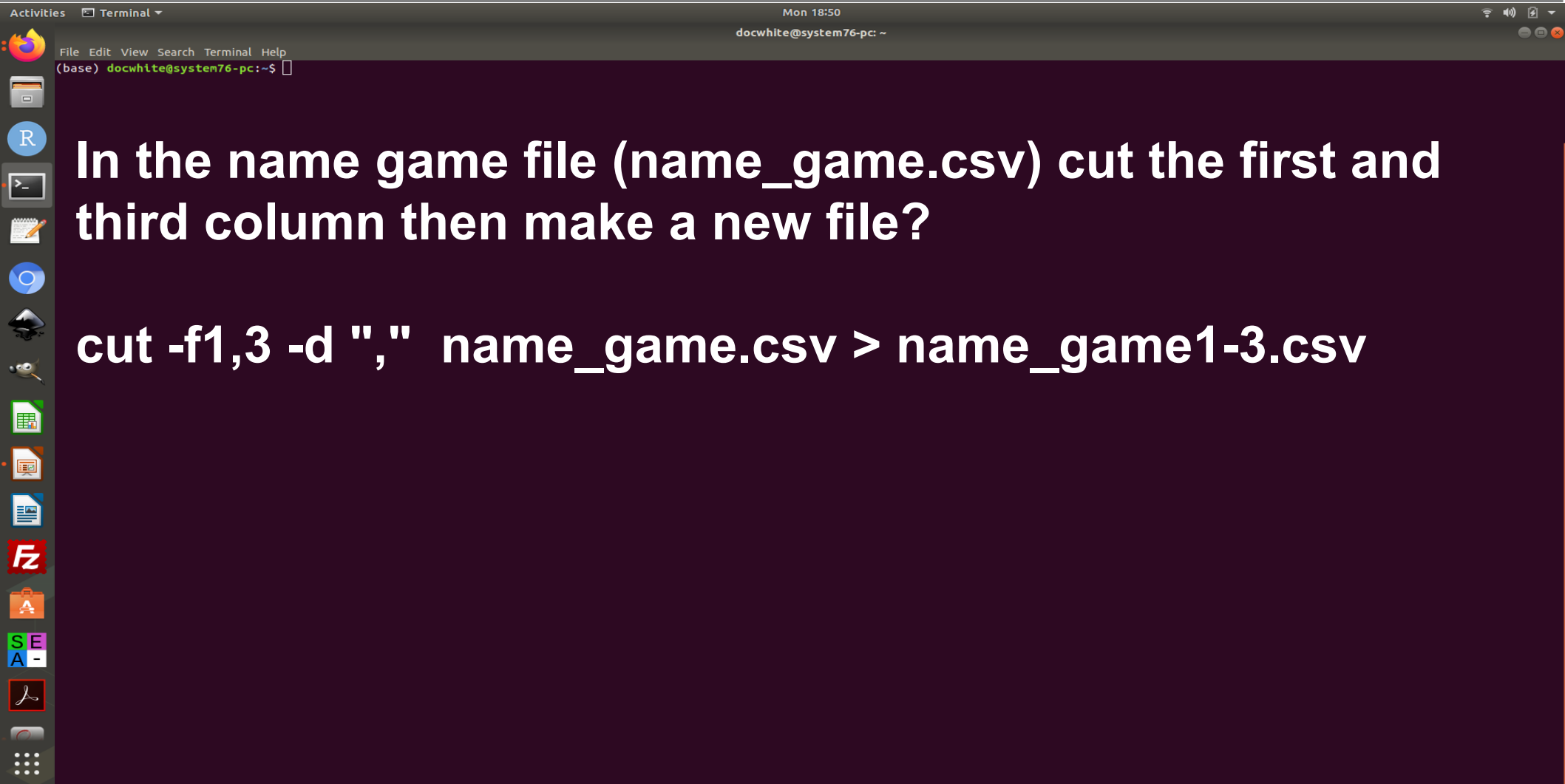
-d (--delimiter) “,”  set field delimiter (default tab)
-f (--fields=LIST) Select by specifying a field
    -f 2 select a field to cut (left is 1)
    -f 2-8,12  select multiple fields to cut
-b (--bytes=LIST) Select by specifying a byte
-c (--characters=LIST) Select by specifying a character
--complement - Complement the selection.
-s (--only-delimited) suppress non-matches

# Cut examples




**In the name game file (name_game.csv) cut the first and third column then make a new file?**

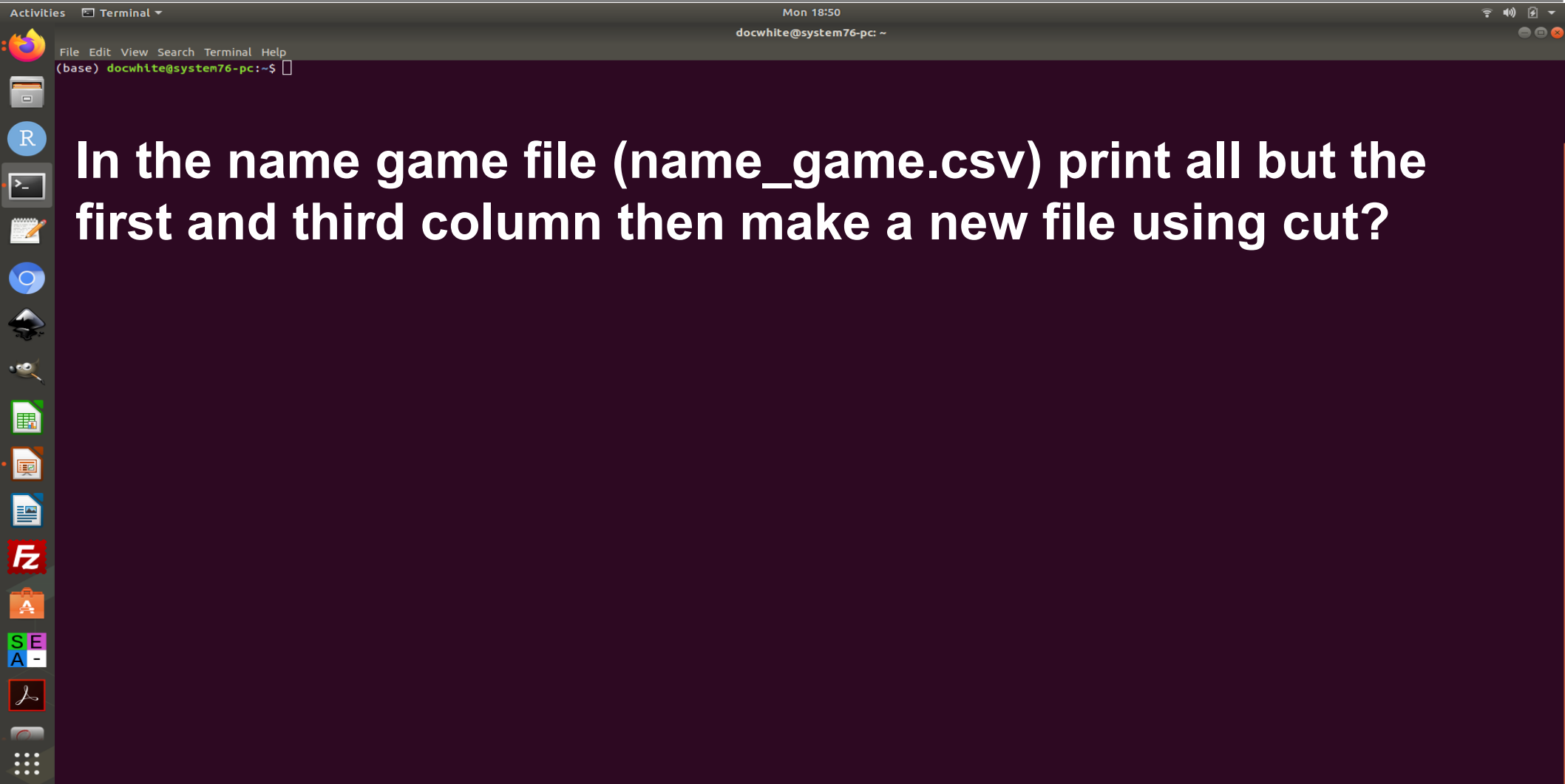
# Cut examples




**In the name game file (name_game.csv) cut the first and third column then make a new file?**

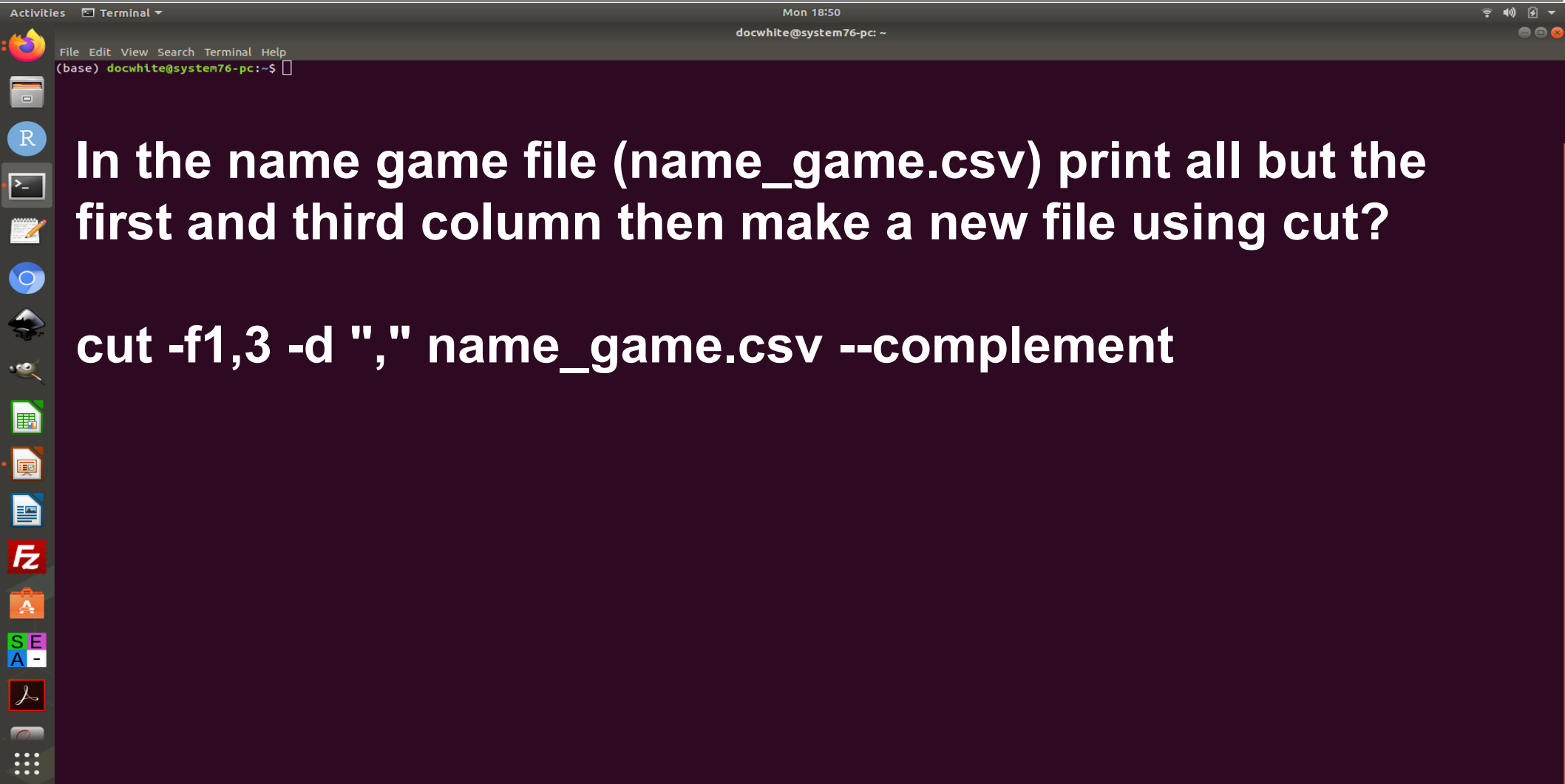
**cut -f1,3 -d "," name_game.csv > name_game1-3.csv**

# Cut examples




**In the name game file (name_game.csv) print all but the first and third column then make a new file using cut?**

# Cut examples




**In the name game file (name_game.csv) print all but the first and third column then make a new file using cut?**

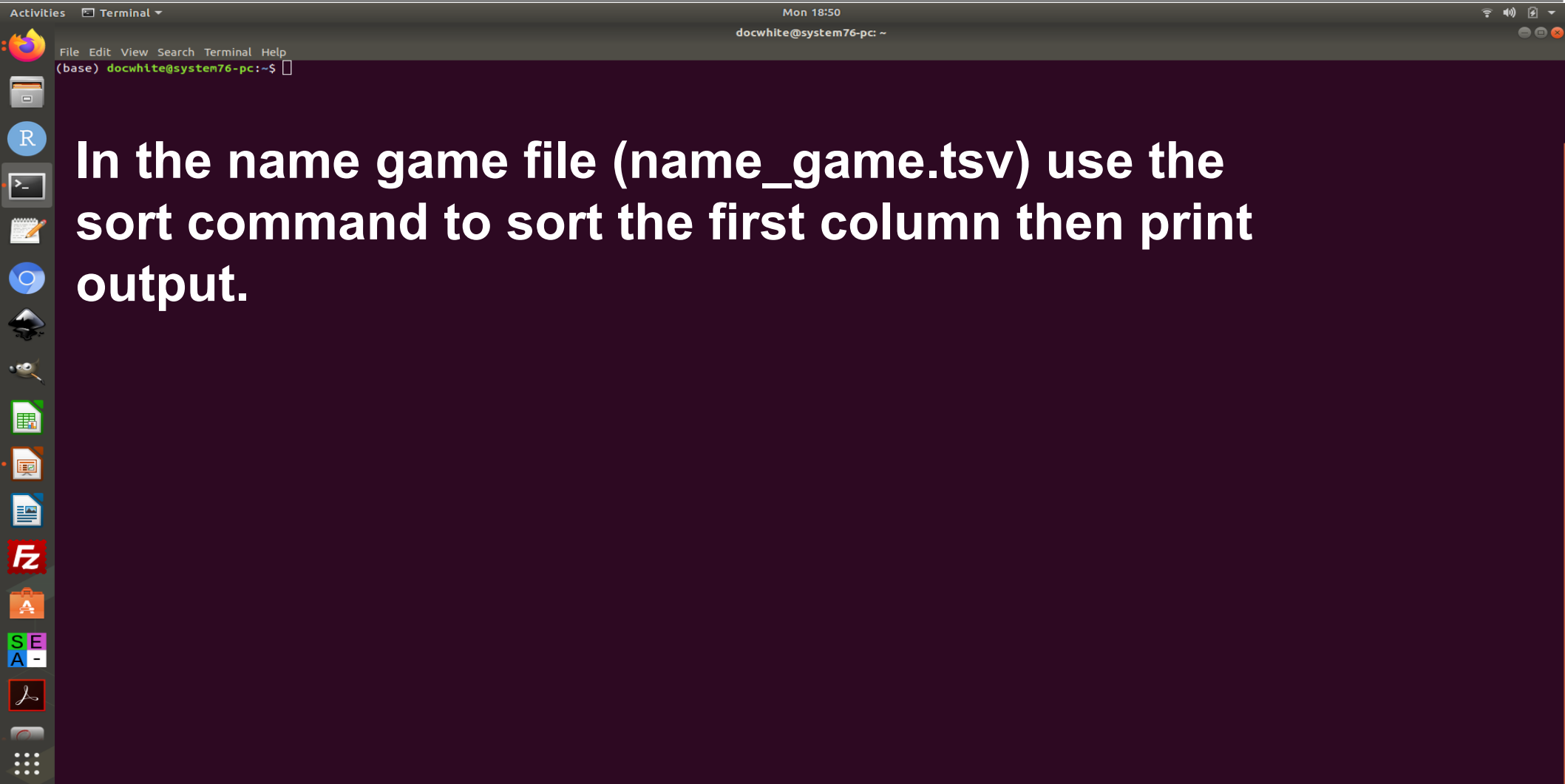
**cut -f1,3 -d "," name_game.csv --complement**

# sort – syntax anatomy of sort

# sort [options] file.txt

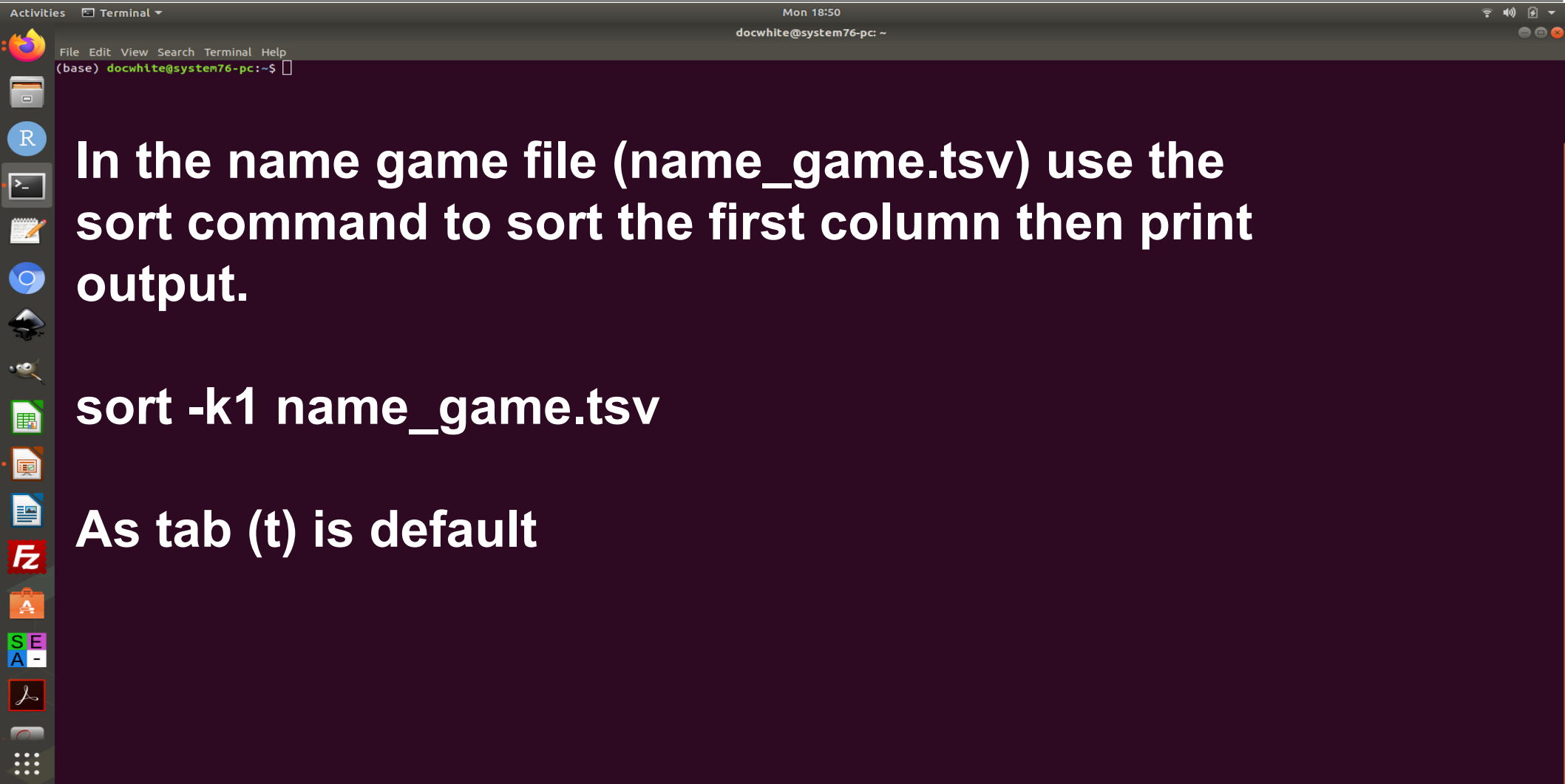
-t “t”  set the delimiter when using -k
default is non-blank to blank transition
-k 3  sort column #3 (left is 1)
-k 2,3  sort multiple columns
-n  sort numerically
-r  reverse sort order
-u  drop duplicates from the result
-b, --ignore-leading-blanks, ignore leading blanks
-d, --dictionary-order consider only blanks and alphanumeric characters
-f, --ignore-case fold lower case to upper case characters
No options: sort alphabetically from leftmost character.

# Sort examples




**In the name game file (name_game.tsv) use the sort command to sort the first column then print output.**

# Sort examples

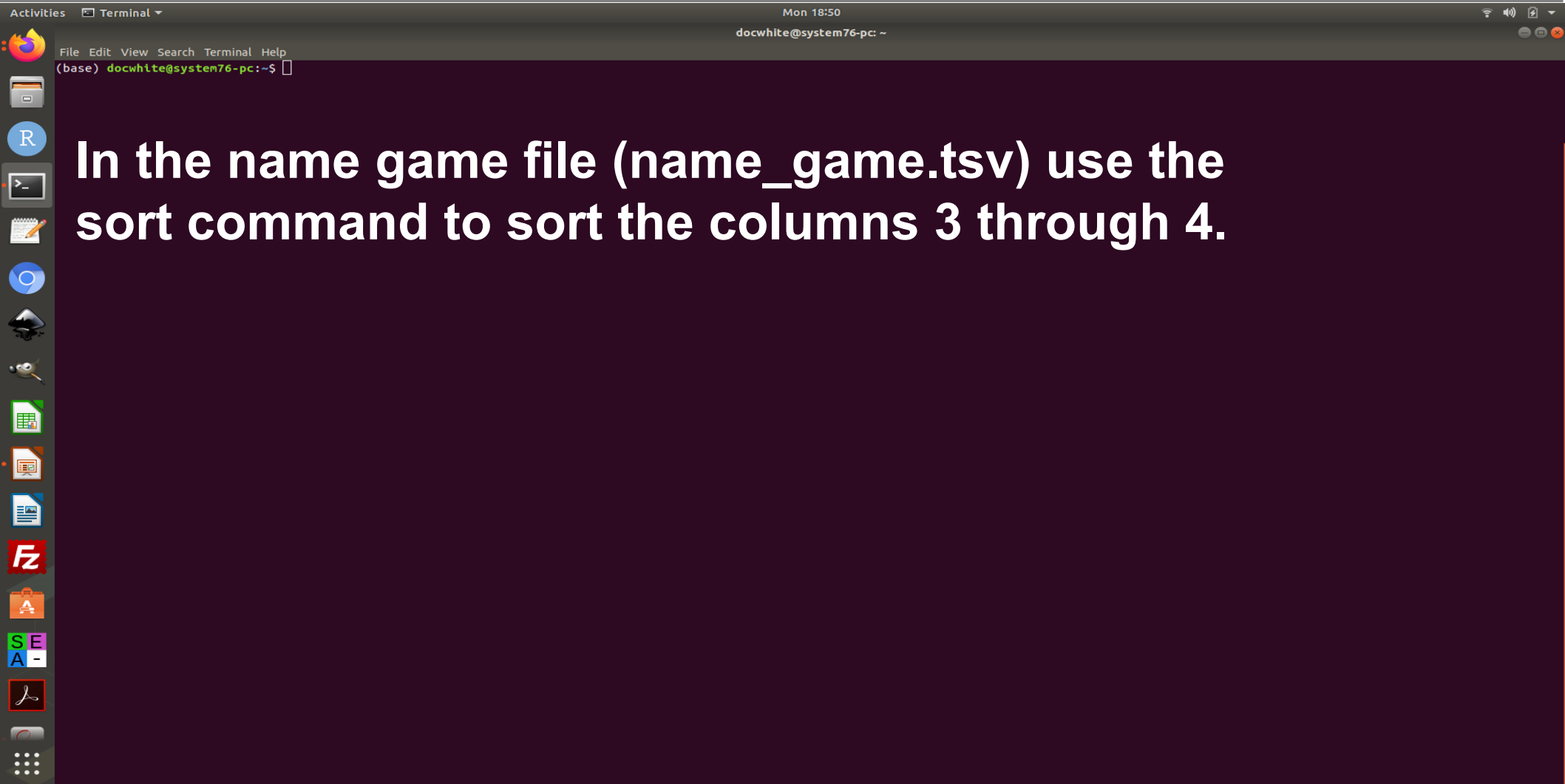



In the name game file (name_game.tsv) use the sort command to sort the first column then print output.

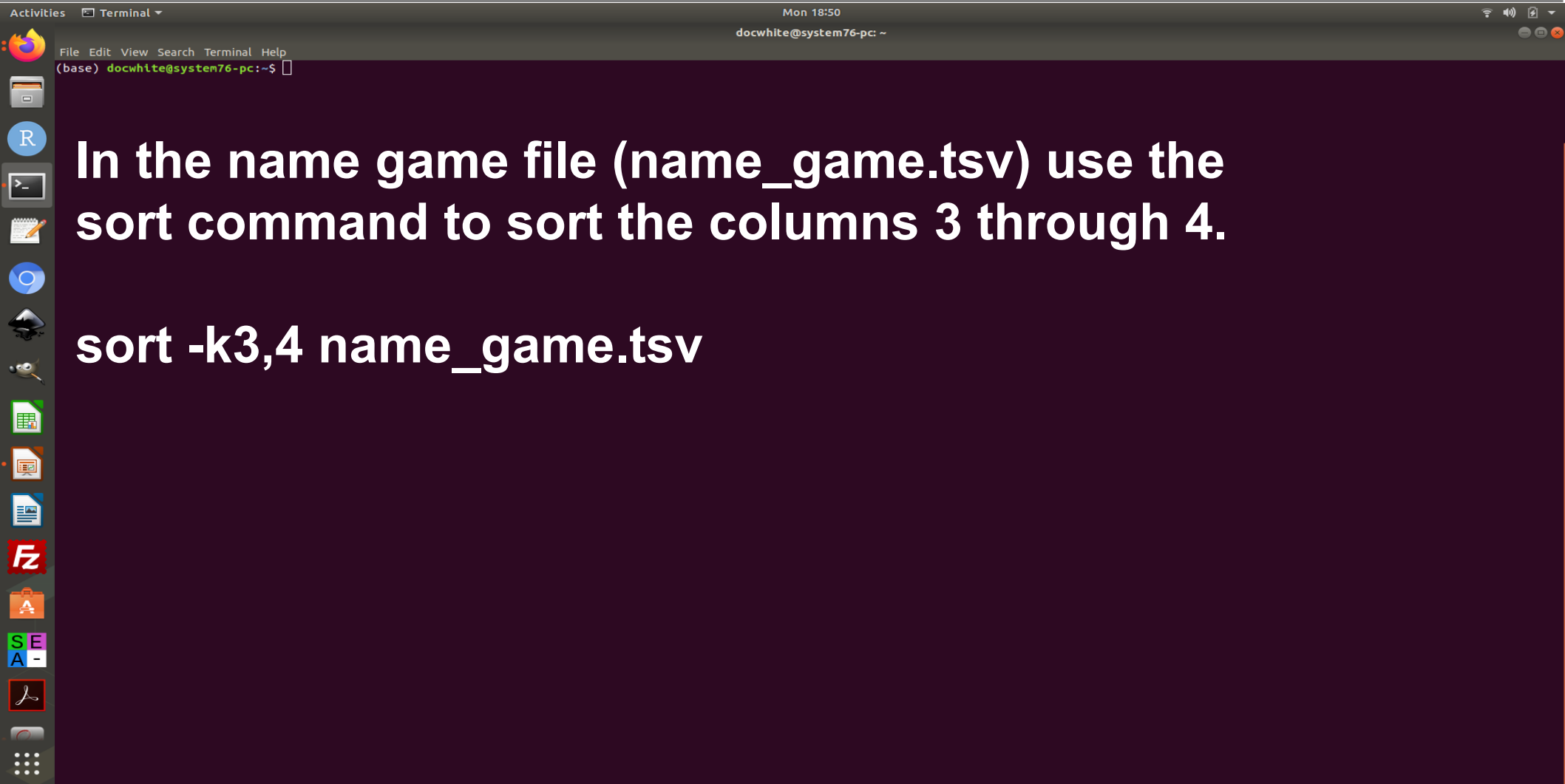
```
sort -k1 name_game.tsv
```

As tab (t) is default

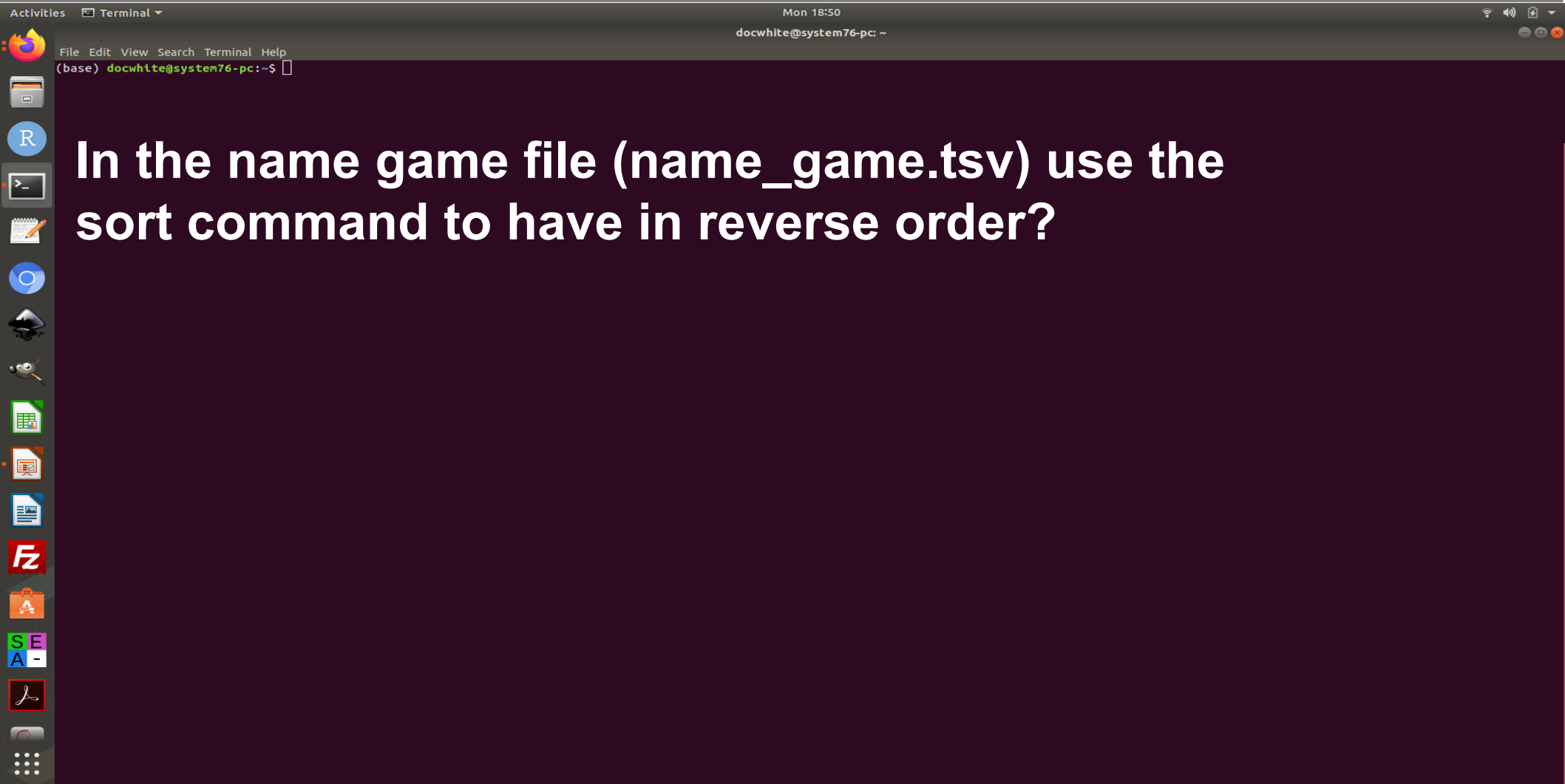
# Sort examples



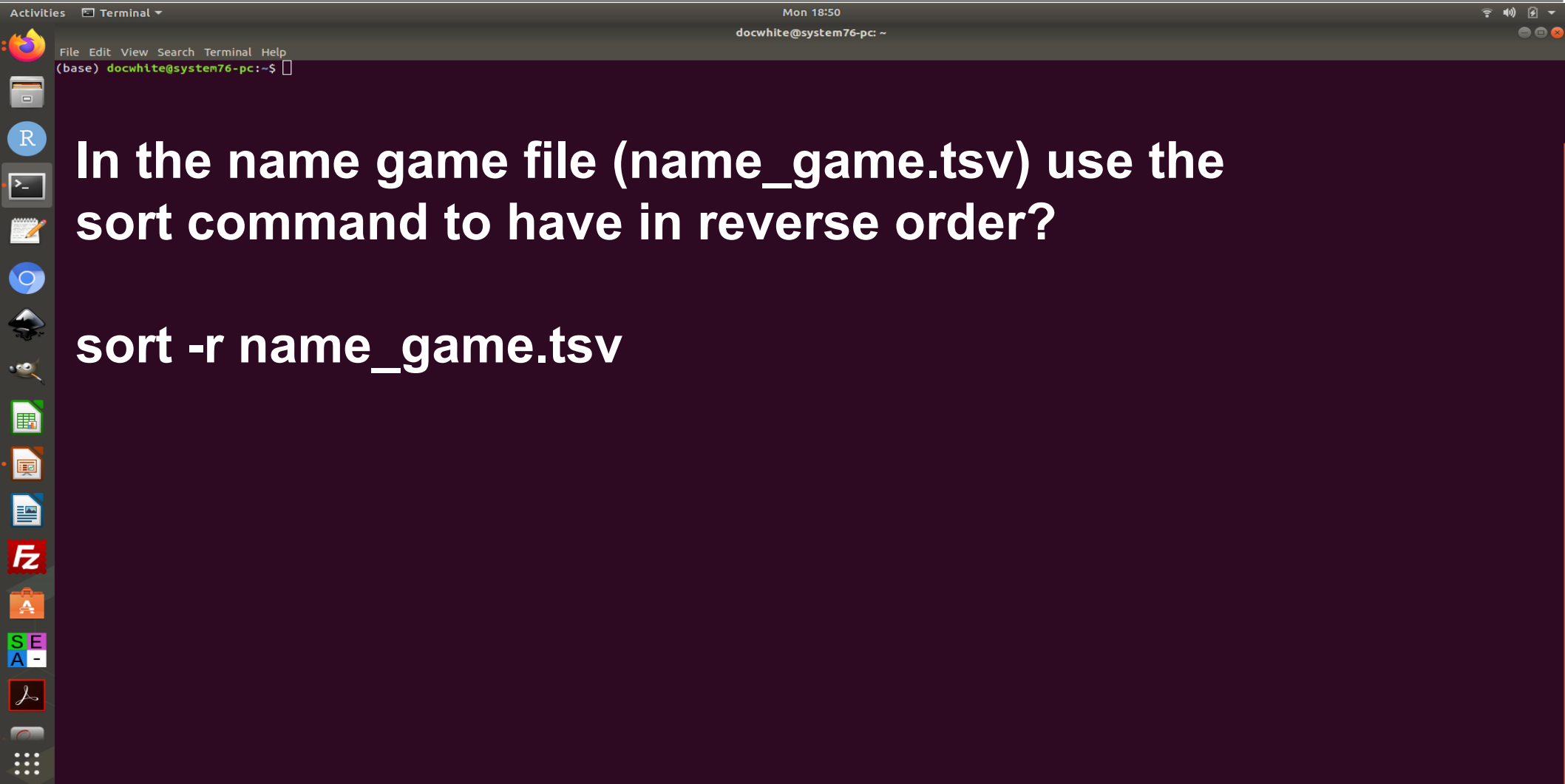
# Sort examples



# Sort examples



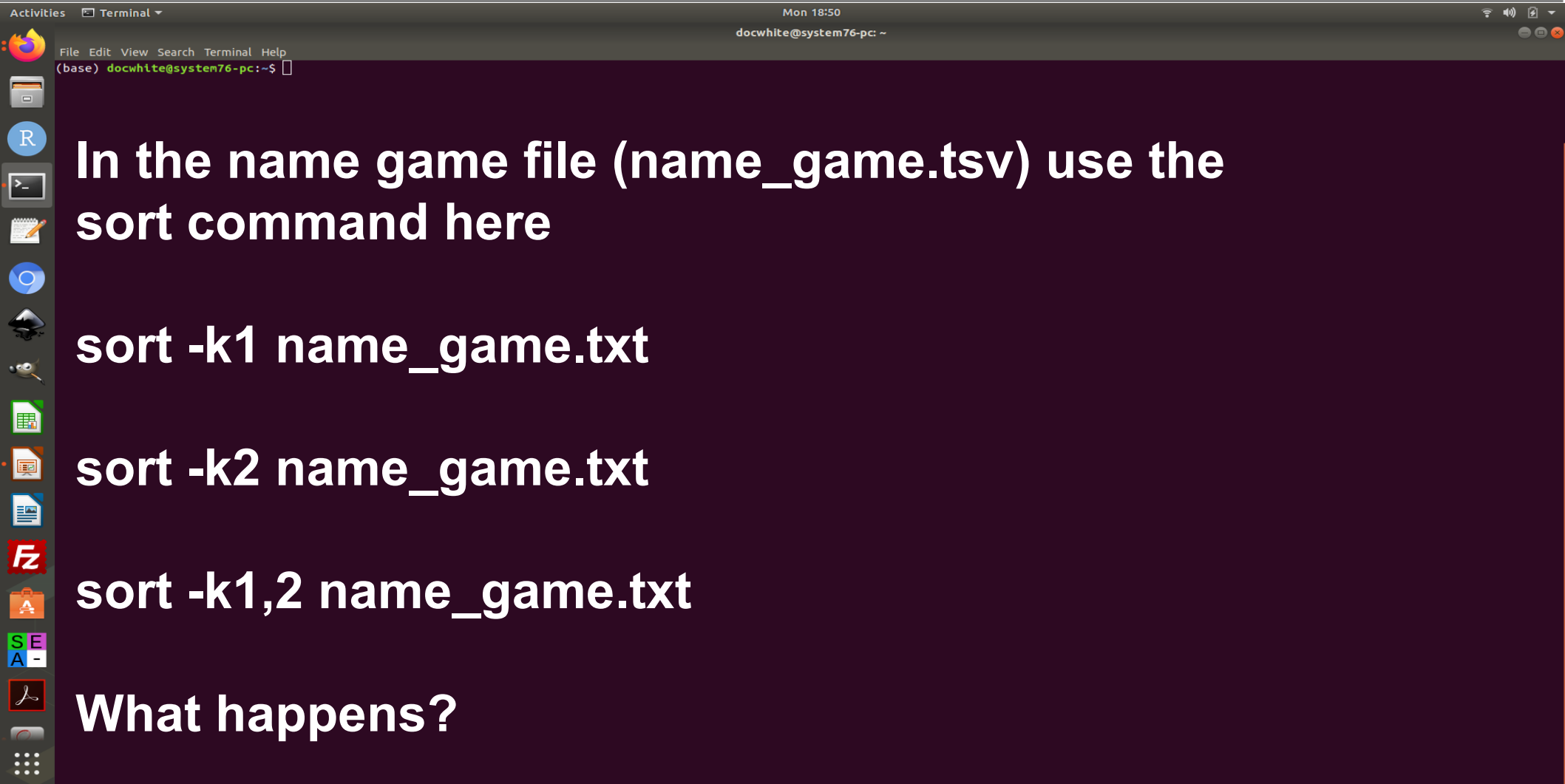
# Sort examples




**In the name game file (name_game.tsv) use the sort command to have in reverse order?**

**sort -r name_game.tsv**

# Sort examples




In the name game file (name_game.tsv) use the sort command here

```
sort -k1 name_game.txt
```

```
sort -k2 name_game.txt
```

```
sort -k1,2 name_game.txt
```

What happens?

# uniq – syntax anatomy of uniq

## uniq [options] file.txt

Only works on sorted files and adjacent lines!

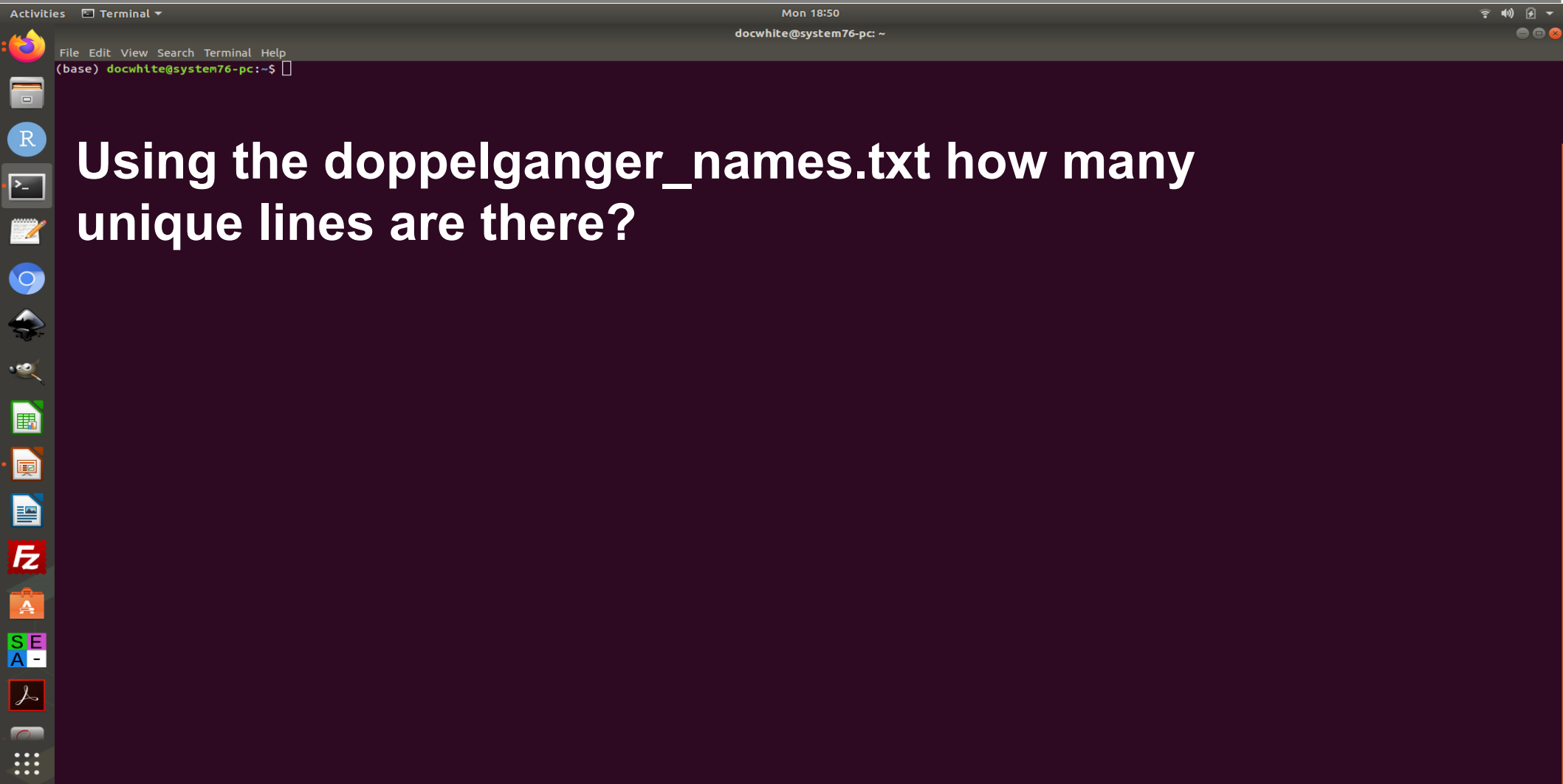
-c count lines for each unique value

-d only report duplicated lines

-u  only report non-duplicated lines

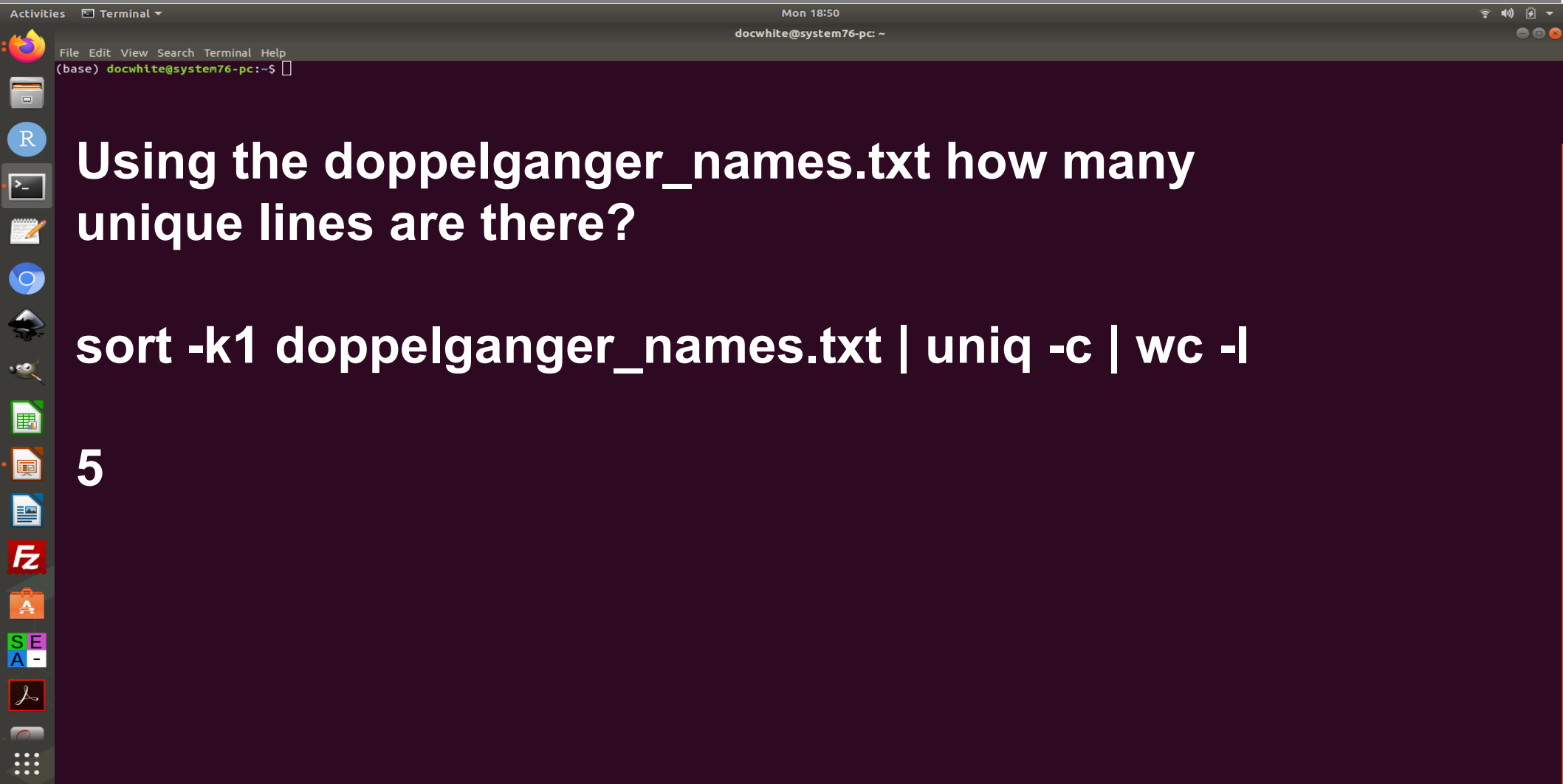
No options: drop all duplicated lines (keeps 1 copy)

# uniq examples




**Using the doppelganger_names.txt how many unique lines are there?**

# uniq examples

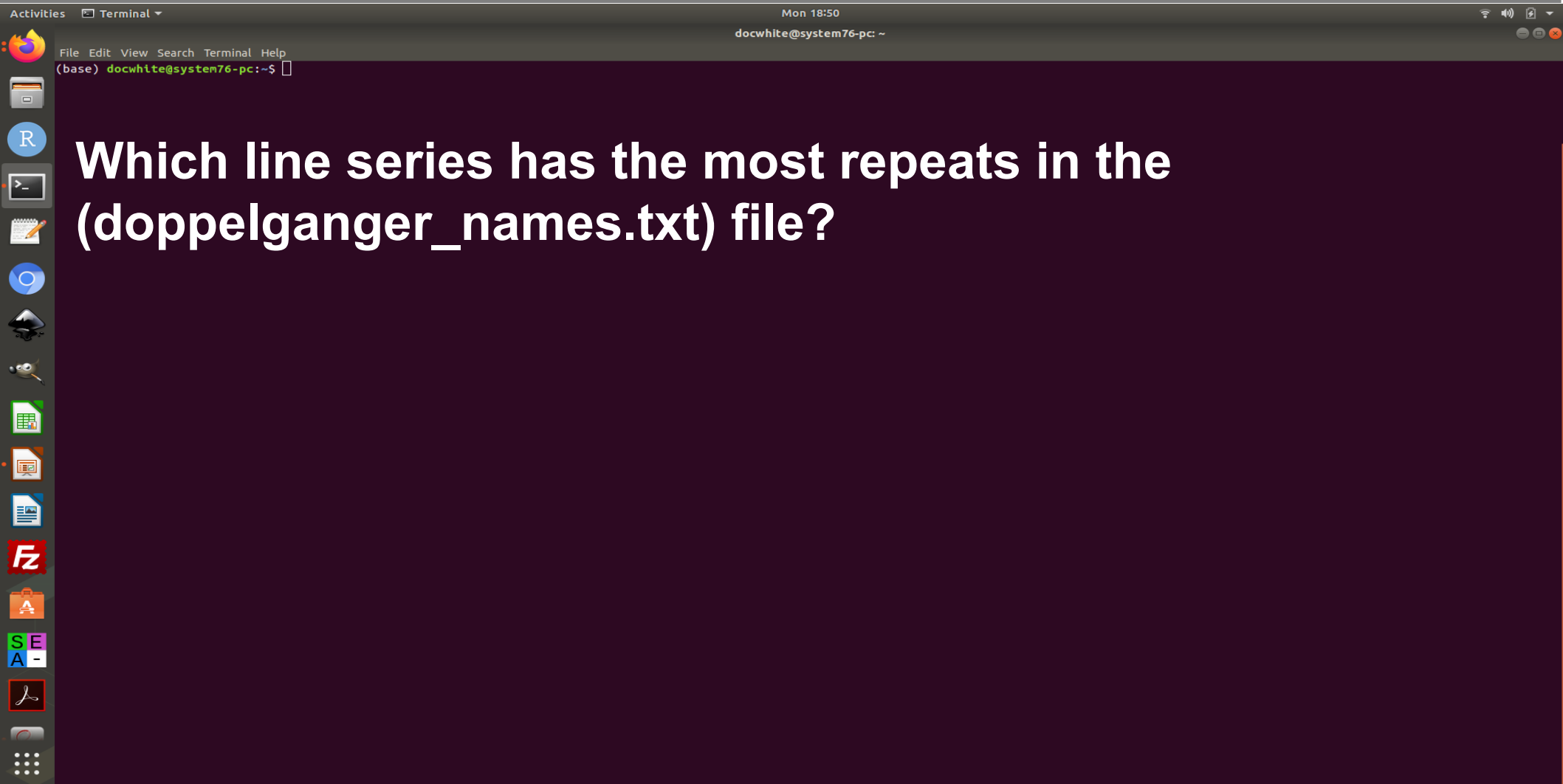



**Using the doppelganger_names.txt how many unique lines are there?**

```
sort -k1 doppelganger_names.txt | uniq -c | wc -l
```

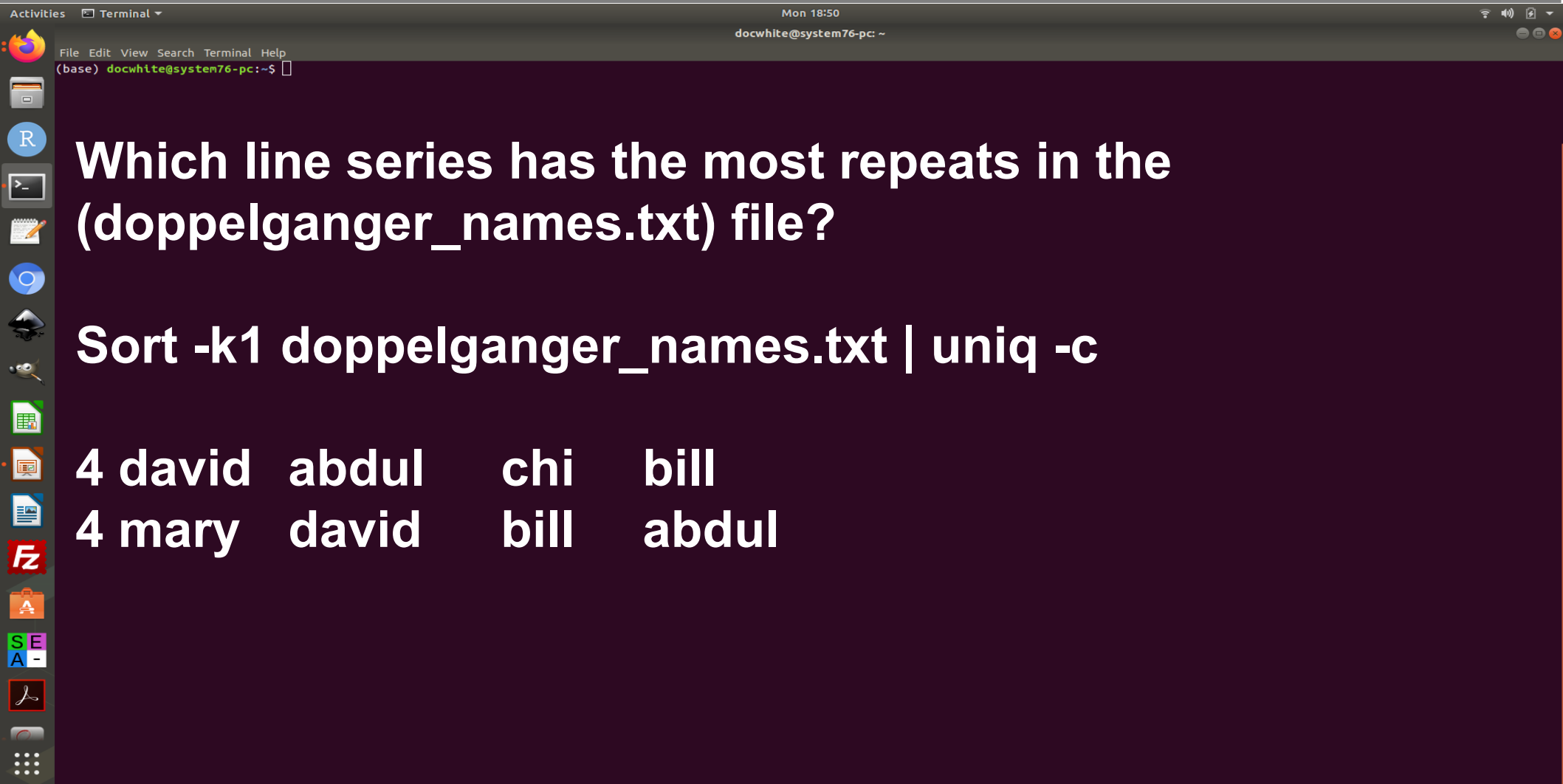
**5**

# uniq examples




**Which line series has the most repeats in the (doppelganger_names.txt) file?**

# uniq examples

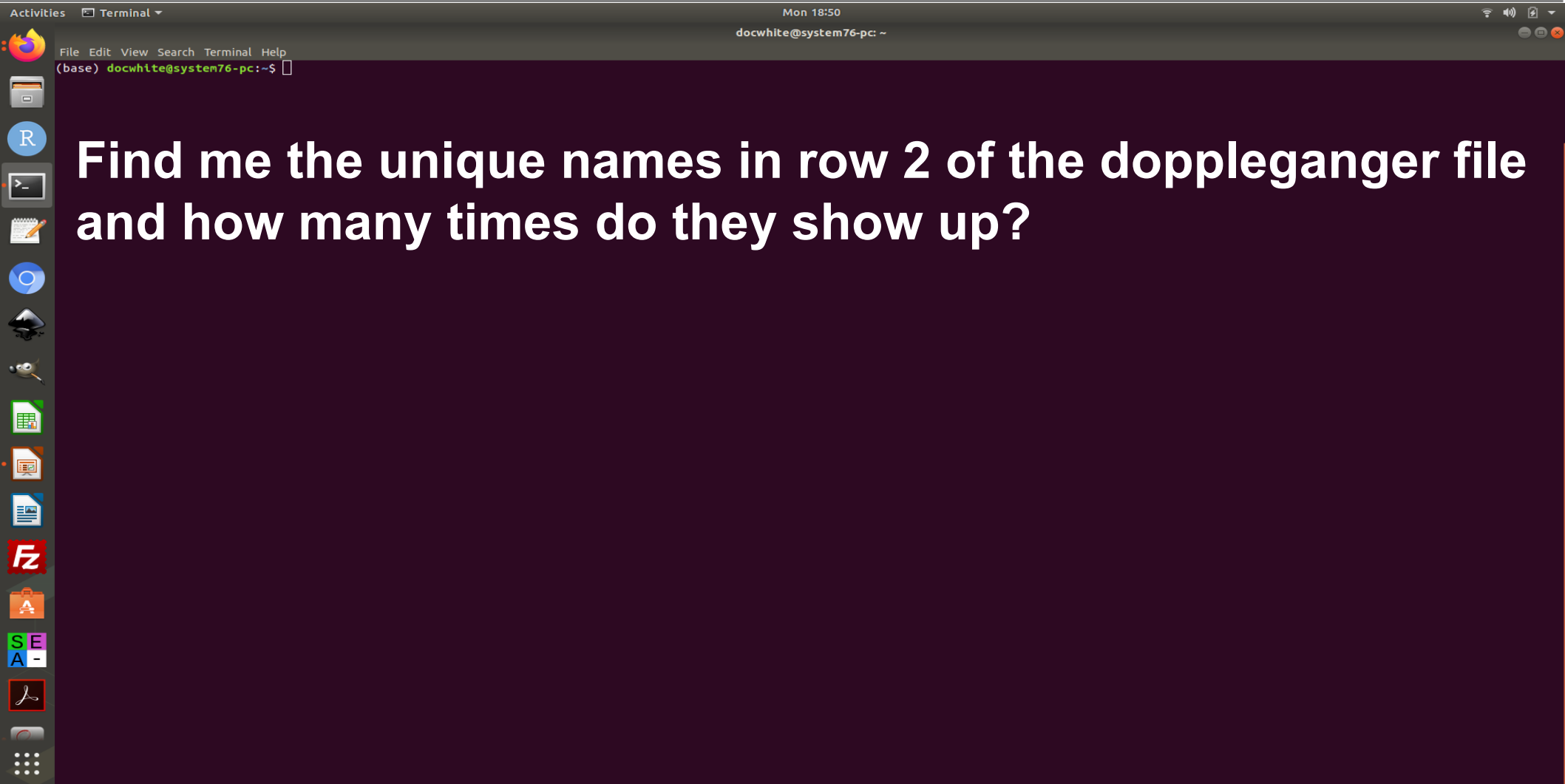



**Which line series has the most repeats in the (doppelganger_names.txt) file?**

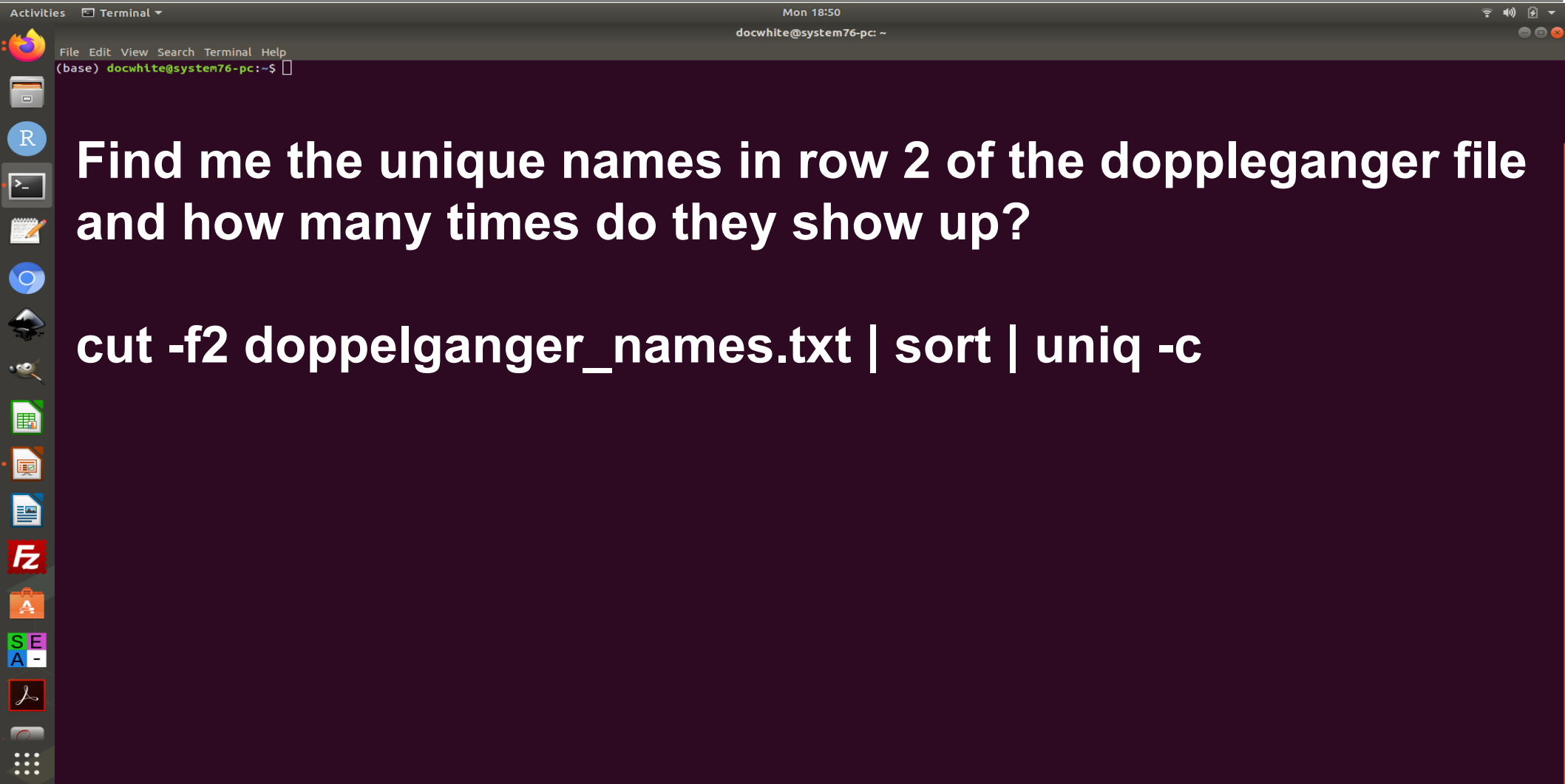
**Sort -k1 doppelganger_names.txt | uniq -c**

**4 david abdul chi bill**
**4 mary david bill abdul**

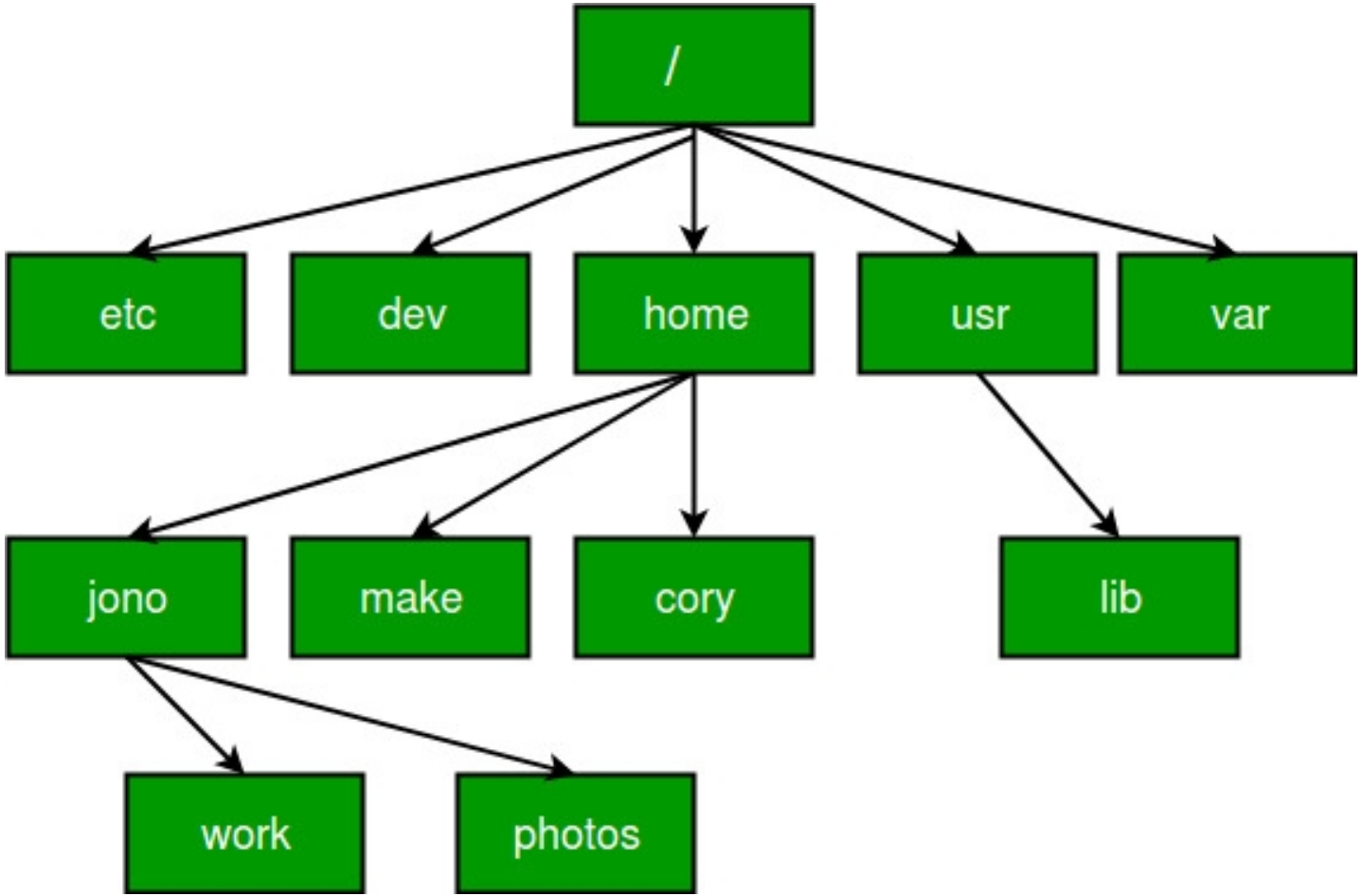
# All together now examples



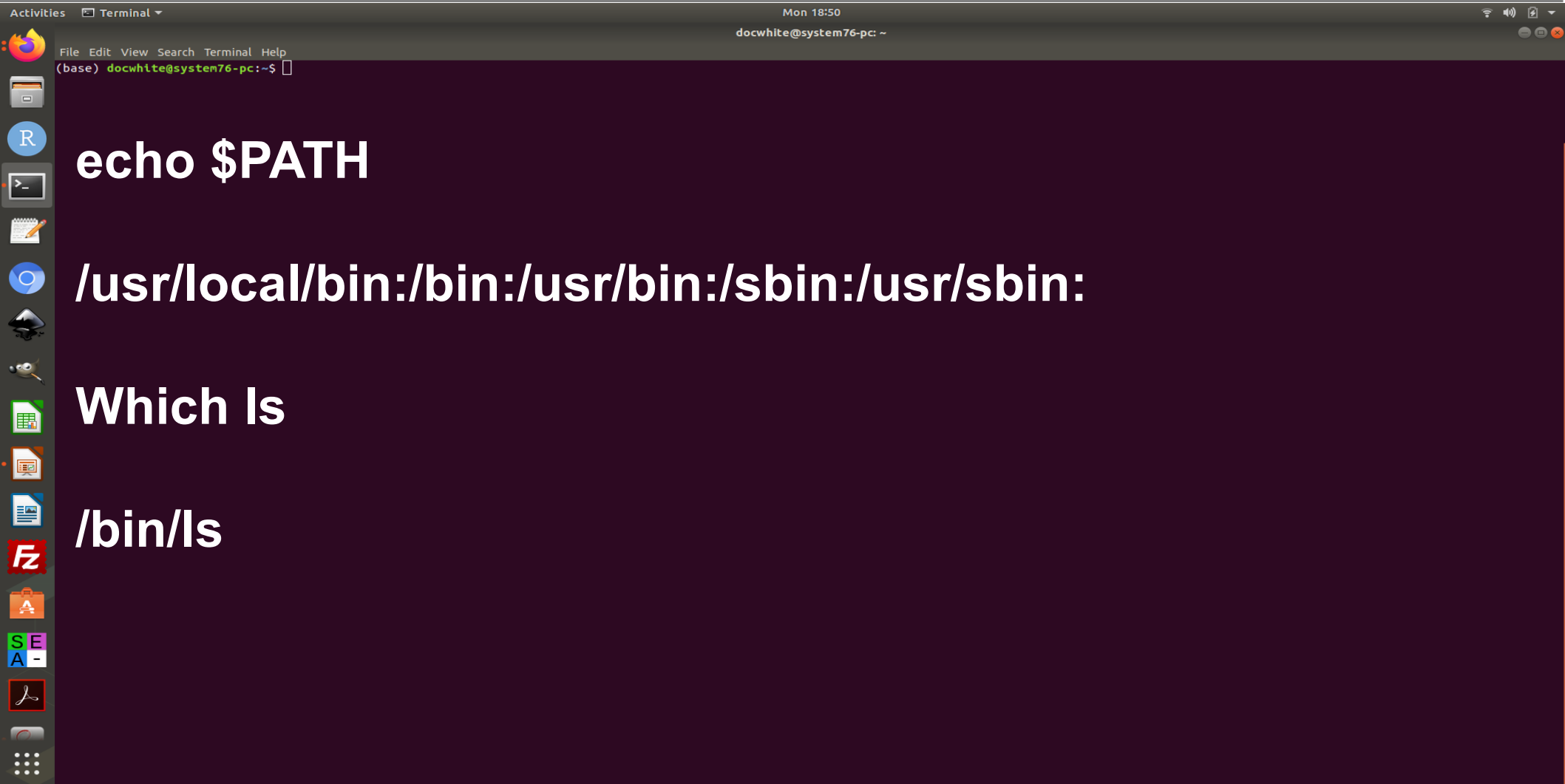
# All together now examples



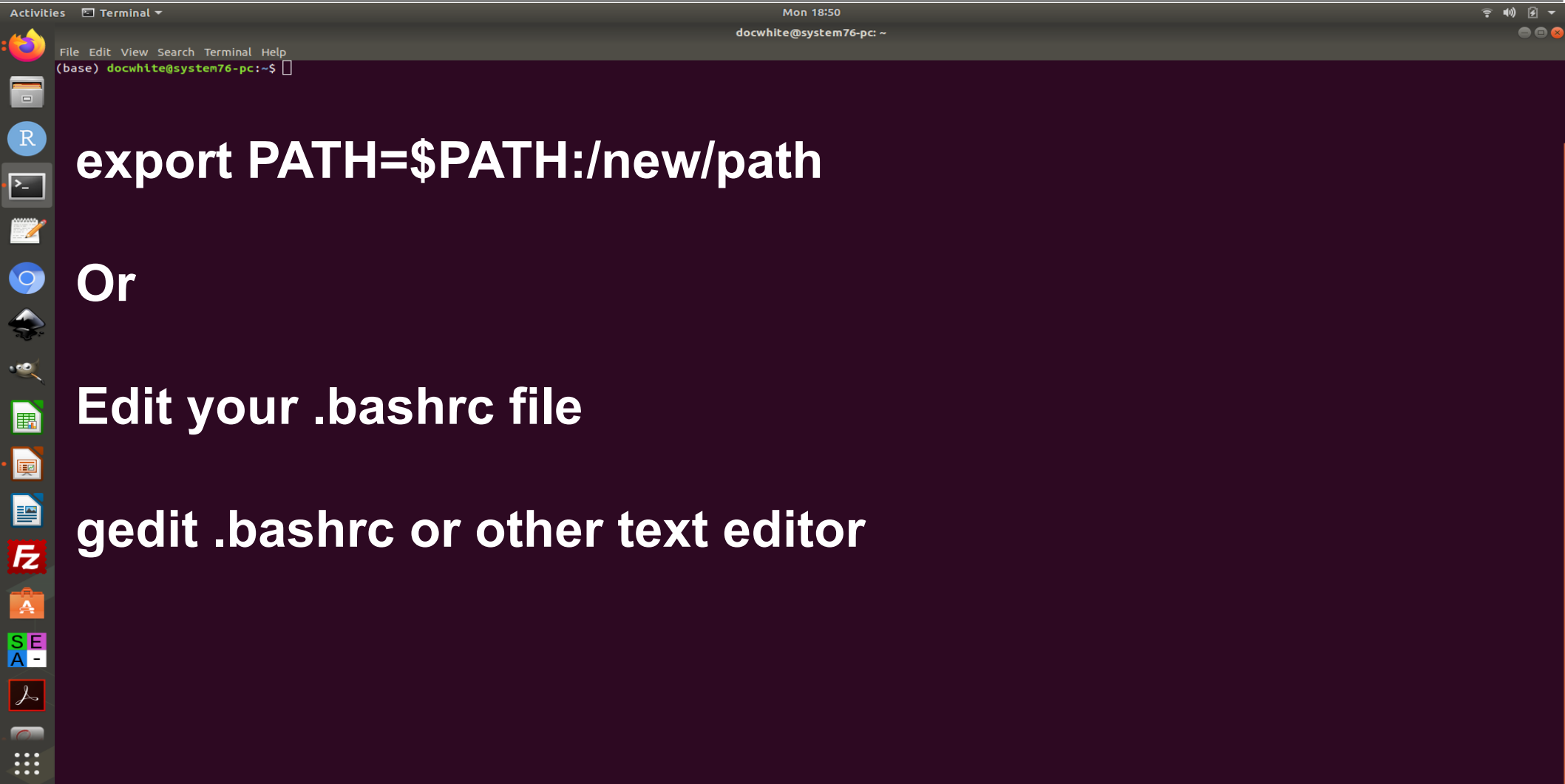
# PATHS



# PATHS


Activities
Terminal
Mon 18:50
docwhite@system76-pc: ~
File Edit View Search Terminal Help
(base) docwhite@system76-pc:~$
echo $PATH
/usr/local/bin:/bin:/usr/bin:/sbin:/usr/sbin:
Which ls
/bin/ls

# PATHS


Activities
Terminal
Mon 18:50
docwhite@system76-pc: ~
File Edit View Search Terminal Help
(base) docwhite@system76-pc:~$
export PATH=$PATH:/new/path
Or
Edit your .bashrc file
gedit .bashrc or other text editor

# Bonus 5

- Convert name game file (name_game.csv) to tsv with:
→ tr
→ awk

# Quiz 5

- On canvas now