VE482
Introduction to Operating Systems

LAB 4

October 17, 2021

Yihua Liu 518021910998

---

1 Database

This is the evening, you are exhausted after a long day of work on `mumsh`, so you decide to poke around and learn more about database, as unfortunately you never had an opportunity to select such course during your studies.

1.1 Database creation

```
git log --pretty="%H,%aN,%aI,%s" > db.csv
```

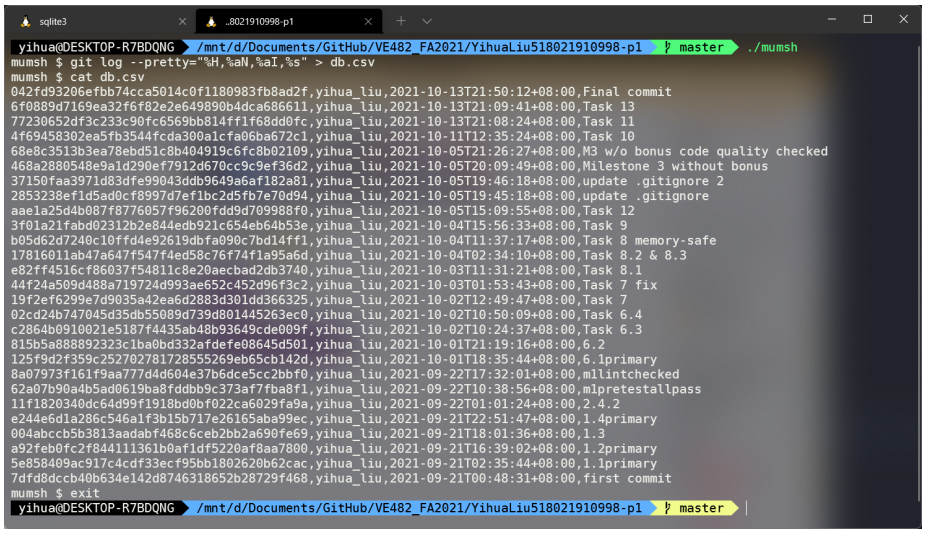

Figure 1. Test database generation.

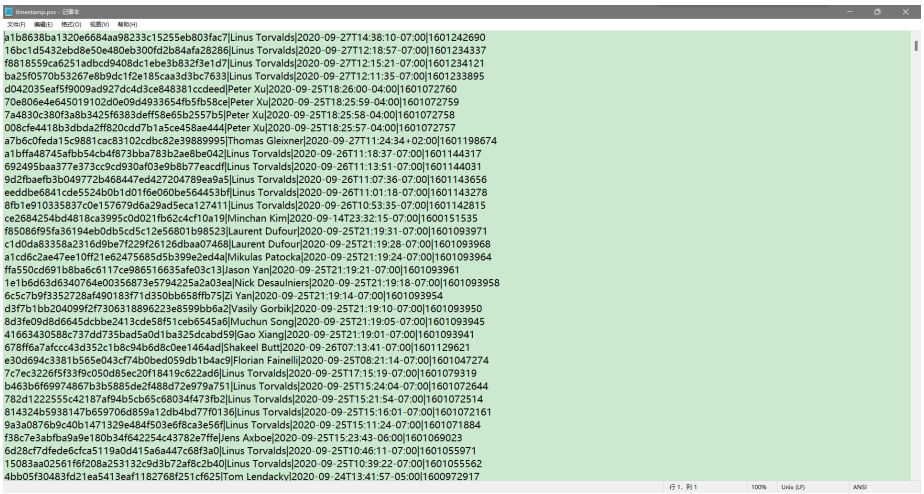

Figure 2. Fields for `timestamp.psv`.

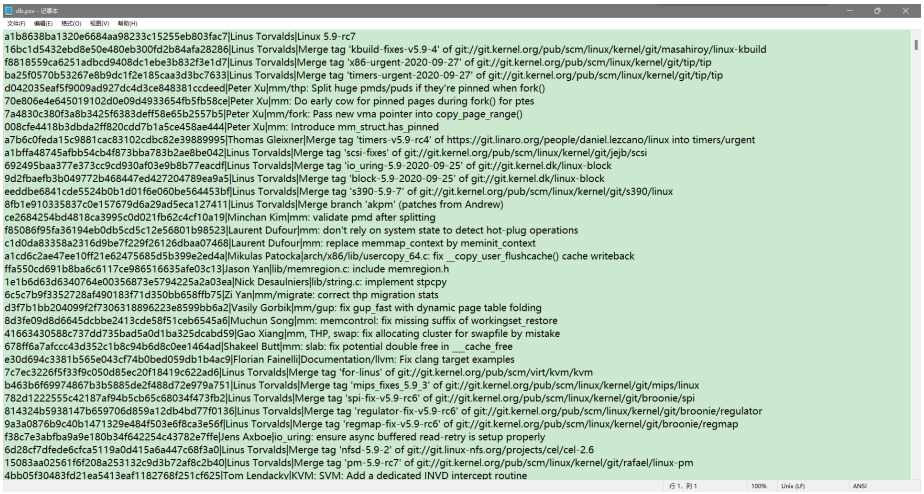

Figure 3. Fields for `db.psv`.

We use `db.psv` and `timestamp.psv` provided by TAs on Canvas instead of `db.csv` and `timestamp.csv` because we cannot find them on the server in the directory `ve482/14`.

1.2 Database system installation

As you want to ensure your understanding and guesses are correct you need to verify a few things online. Luckily your VPN is now working, so you can use a proper search engine and ensure the correctness of what you found.

- What are the most common database systems?
  Oracle, MySQL, Microsoft SQL Server, MongoDB, Redis, SQLite, Microsoft Access, etc.

- Briefly list the pros and cons of the three most common ones.

Oracle	Pros	Robust, strong and abundant features, state-of-the-art
	Cons	Expensive, occupying much resources
MySQL	Pros	Available for free, extensible, high-customization
	Cons	Complex to use, some features are missing
Microsoft SQL Server	Pros	Robust, strong and abundant features, fast and stable
	Cons	Very expensive, occupying much resources

Table 1. The pros and cons of the three most common database systems [1].

After completing your reading you decide to install `SQLite` on your Linux system. The next step is now to import your git database into two tables.

- Create an empty SQLite database.

```
sudo apt install sqlite3
sqlite3 l4.db
```

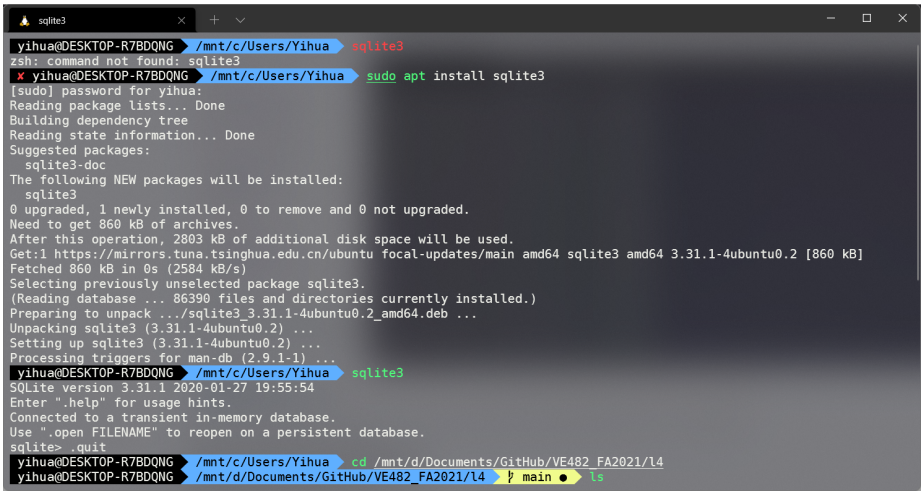

Figure 4. Create an empty SQLite database.

- Use the SQLite shell to prepare two empty tables for each of your .csv file.

```
CREATE TABLE db
(
    hash TEXT NOT NULL,
    name TEXT NOT NULL,
    comment TEXT NOT NULL
);
```

```
CREATE TABLE time_stamp
(
    hash TEXT NOT NULL,
    name TEXT NOT NULL,
    dates TEXT,
    tstamp INT
);
```

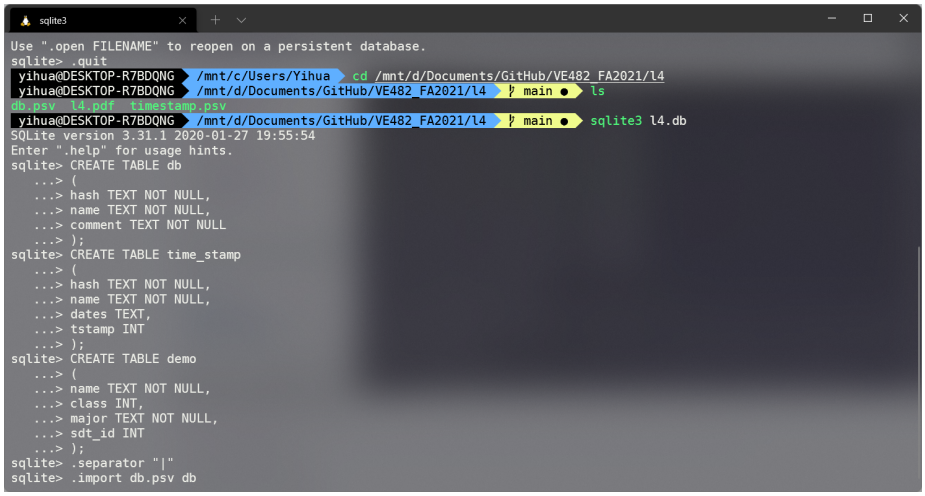

Figure 5. Use the SQLite shell to prepare two empty tables for each of your .csv file.

- Import each .csv file in its corresponding SQLite table.

```
.separator "|"
.import db.psv db
.import timestamp.psv time_stamp
```

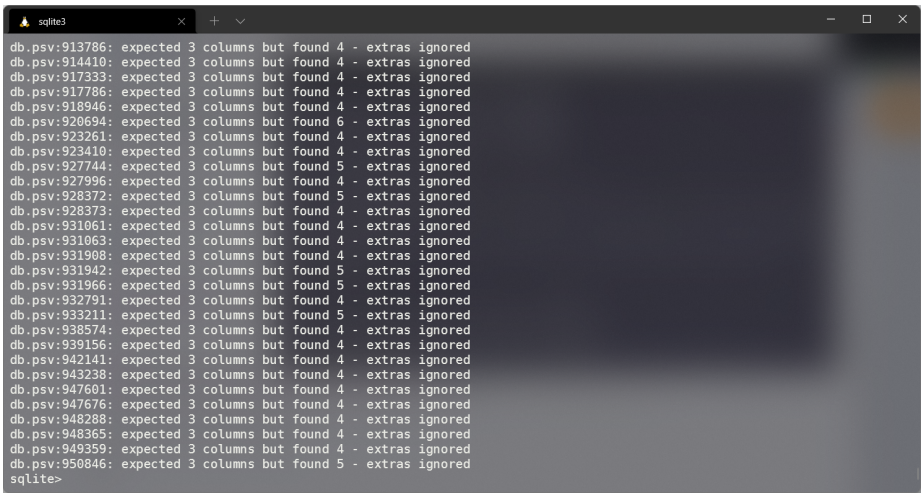

Figure 6. Import db.psv in its corresponding SQLite table.

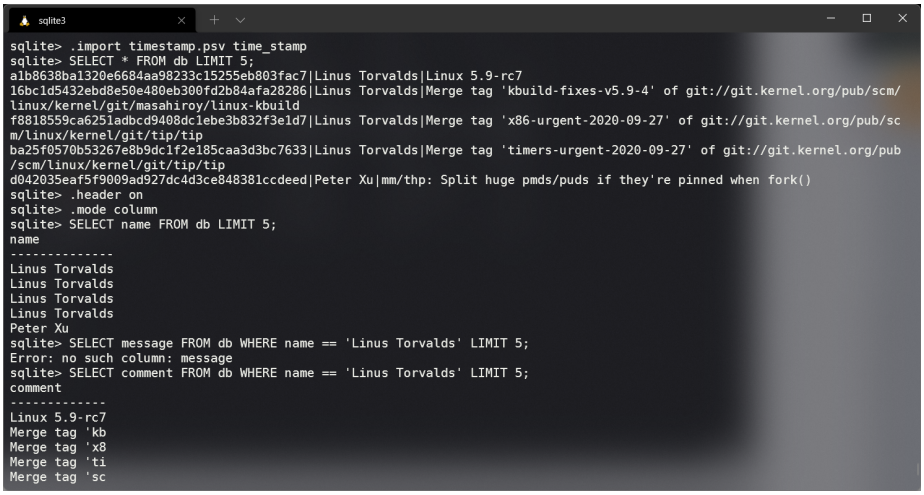

Figure 7. Import timestamp.psv in its corresponding SQLite table.

1.3 Database queries

At this stage you want to run basic queries to verify that the database has been imported correctly. Therefore you spend the rest of the evening playing around the database and running queries.

- Who are the top five contributors to the Linux kernel since the beginning?

```
SELECT name, COUNT(name) FROM time_stamp GROUP BY name
↪  ORDER BY COUNT(name) DESC LIMIT 5;
```

```
Linus Torvalds|30702
David S. Miller|13180
Takashi Iwai|7726
Mark Brown|7670
Arnd Bergmann|7520
```

- Who are the top five contributors to the Linux kernel for each year over the past five years?

```
SELECT name, COUNT(name) FROM time_stamp WHERE dates
↪  BETWEEN DATETIME('now','-5 year') AND
↪  DATETIME('now','-4 year') GROUP BY name ORDER BY
↪  COUNT(name) DESC LIMIT 5;
SELECT name, COUNT(name) FROM time_stamp WHERE dates
↪  BETWEEN DATETIME('now','-4 year') AND
↪  DATETIME('now','-3 year') GROUP BY name ORDER BY
↪  COUNT(name) DESC LIMIT 5;
SELECT name, COUNT(name) FROM time_stamp WHERE dates
↪  BETWEEN DATETIME('now','-3 year') AND
↪  DATETIME('now','-2 year') GROUP BY name ORDER BY
↪  COUNT(name) DESC LIMIT 5;
SELECT name, COUNT(name) FROM time_stamp WHERE dates
↪  BETWEEN DATETIME('now','-2 year') AND
↪  DATETIME('now','-1 year') GROUP BY name ORDER BY
↪  COUNT(name) DESC LIMIT 5;
SELECT name, COUNT(name) FROM time_stamp WHERE dates
↪  BETWEEN DATETIME('now','-1 year') AND DATETIME('now')
↪  GROUP BY name ORDER BY COUNT(name) DESC LIMIT 5;
```

From 5 years ago to 4 years ago:

```
Linus Torvalds|2292
David S. Miller|1365
Chris Wilson|1085
Arnd Bergmann|1016
Arvind Yadav|764
```

From 4 years ago to 3 years ago:

```
Linus Torvalds|2113
David S. Miller|1426
Arnd Bergmann|1147
Colin Ian King|831
Chris Wilson|812
Arvind Yadav|764
```

From 3 years ago to 2 years ago:

```
Linus Torvalds|2440
David S. Miller|1241
Christoph Hellwig|967
YueHaibing|935
Chris Wilson|910
```

From 1 years ago to 1 year ago:

```
Linus Torvalds|2358
David S. Miller|1166
Christoph Hellwig|1001
Chris Wilson|985
Mauro Carvalho Chehab|771
```

From 1 year ago to now:

```

```

This is empty because the data only records till 2020-09-27T14:38:10-07:00. In MySQL, you can use sentences like `FROM_UNIXTIME('tstamp') > DATE_SUB(NOW(), INTERVAL 5 YEAR)` or `tstamp > UNIX_TIMESTAMP(DATE_SUB(NOW(), INTERVAL 5 YEAR))` [2].

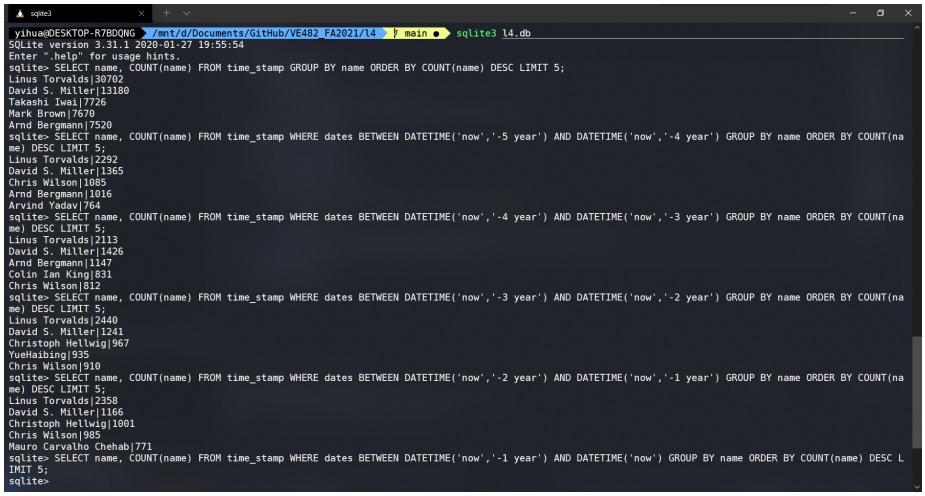

Figure 8. The top five contributors to the Linux kernel since the beginning and each year over the last 5 years.

- What is the most common "commit subject"?

```
SELECT comment, COUNT(comment) FROM db GROUP BY comment
↪  ORDER BY COUNT(comment) DESC LIMIT 1;
```

```
Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem
↪  /net|670
```

- On which day is the number of commits the highest?

```
SELECT DATE(dates), count(DATE(dates)) FROM time_stamp
↪  GROUP BY DATE(dates) ORDER BY count(DATE(dates)) DESC
↪  LIMIT 5;
```

```
2008-01-30|1031
```

- Determine the average time between two commits for the five main contributor.

```
SELECT (MAX(tstamp) - MIN(tstamp)) /
↪  (COUNT(DISTINCT(tstamp)) - 1) FROM time_stamp WHERE
↪  name = 'Linus Torvalds';
SELECT (MAX(tstamp) - MIN(tstamp)) /
↪  (COUNT(DISTINCT(tstamp)) - 1) FROM time_stamp WHERE
↪  name = 'David S. Miller';
SELECT (MAX(tstamp) - MIN(tstamp)) /
↪  (COUNT(DISTINCT(tstamp)) - 1) FROM time_stamp WHERE
↪  name = 'Takashi Iwai';
SELECT (MAX(tstamp) - MIN(tstamp)) /
↪  (COUNT(DISTINCT(tstamp)) - 1) FROM time_stamp WHERE
↪  name = 'Mark Brown';
SELECT (MAX(tstamp) - MIN(tstamp)) /
↪  (COUNT(DISTINCT(tstamp)) - 1) FROM time_stamp WHERE
↪  name = 'Arnd Bergmann';
```

```
15886
36972
63730
63205
64693
```

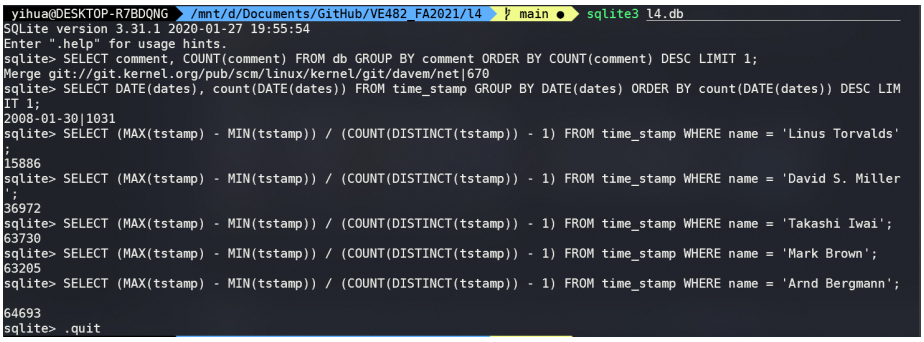

Figure 9. The results of the last three questions.

The time differences are measured in seconds. Note that Microsoft Access does not support syntax `COUNT(DISTINCT column_name)`. In MySQL, you can use function `TIMESTAMPDIFF()` to calculate them more flexibly [3].

2 Debugging

You are pretty happy and enjoying the database tasks when your mum pops in your room. She looks pretty upset that you are still not asleep as she thinks you were playing video games...

When you explain her to that you have terrible bugs in your shell and needed a bit of change, she asked you whether you had used GDB. As you replied "Can I eat it?" she realises you probably do not know much about it. She kindly tells you to have a quick try at it on your current `mumsh` version to preview it. This should become very handy if you ever have to work on a large scale project.

- How to enable built-in debugging in `gcc`?
  Adding a `-g` option. It can produce debugging information in the operating system's native format (stabs, COFF, XCOFF, or DWARF). GDB can work with this debugging information [5].

- What is the meaning of GDB?
  GDB is the GNU Project debugger that allows you to see what is going on 'inside' another program while it executes – or what another program was doing at the moment it crashed, and can do four main kinds of things (plus other things in support of these) to help you catch bugs [9].

- Compile the master branch of your `mumsh` with debugging enabled.

```
gcc -std=gnu11 -O2 -Wall -Wextra -Werror -pedantic
↪  -Wno-unused-result -Wconversion
↪  -fno-omit-frame-pointer -fsanitize=undefined
↪  -fsanitize=address -o mumsh p1.c -g
```

2.1 Basic GDB usage

- Find the homepage of the GDB project [9].
  https://www.gnu.org/software/gdb
- What languages are supported by GDB?
  GDB supports the following languages (in alphabetical order) [9]:
  - Ada
  - Assembly
  - C
  - C++
  - D
  - Fortran
  - Go
  - Objective-C
  - OpenCL
  - Modula-2
  - Pascal
  - Rust

```
(gdb) file mumsh
(gdb) break p1.c:17
(gdb) break strqtok
(gdb) run
mumsh $ echo 123
(gdb) next
(gdb) step
(gdb) watch delim_offset
(gdb) print delim_offset
(gdb)
(gdb) continue
(gdb) continue
(gdb) continue
mumsh $ exit
(gdb) continue
(gdb) continue
```

Figure 10. Working with GDB.

- What are the following GDB commands doing [8]:
  - `backtrace`: (also `bt`) A backtrace is a summary of how your program got where it is. It shows one line per frame, for many frames, starting with the currently executing frame (frame zero), followed by its caller (frame one), and on up the stack [7].
  - `delete`: `delete checkpoint` *`checkpoint-id`*: Delete the previously-saved checkpoint identified by *checkpoint-id*.
    `delete [breakpoints] [`*`list...`*`]`: Delete the breakpoints, watchpoints, or catchpoints of the breakpoint list specified as argument.
    `delete display` *`dnums...`*: Remove items from the list of expressions to display.
    `delete mem` *`nums...`*: Remove memory regions *nums...* from the list of regions monitored by gdb.
    `delete tracepoint [`*`nums...`*`]`: Permanently delete one or more tracepoints.

`delete tvariable [$name...]`: Delete the given trace state variables, or all of them if no arguments are specified.

- `where`: The name `where` is additional alias for `backtrace`.
- `info breakpoints`: (also `info break`) Print a table of all breakpoints, watchpoints, and catchpoints set and not deleted.
- `finish`: (also `fin`) Continue running until just after function in the selected stack frame returns. Print the returned value (if any).

- Search the documentation and explain how to use conditional breakpoints [8].
  A breakpoint with a condition evaluates the expression each time your program reaches it, and your program stops only if the condition is `true`.
  `condition` *`bnum expression`*: Specify expression as the break condition for breakpoint, watchpoint, or catchpoint number *bnum*.
  `condition -force` *`bnum expression`*: When the `-force` flag is used, define the condition even if *expression* is invalid at all the current locations of breakpoint *bnum*.
  `condition` *`bnum`*: Remove the condition from breakpoint number *bnum*. It becomes an ordinary unconditional breakpoint.
  Watch this youtube video and answer the following questions.

- What is `-tui` option for GDB?
  `tui reg float`: show floating registers.
  The GDB Text User Interface (TUI) is a terminal interface which uses the `curses` library to show the source file, the assembly output, the program registers and GDB commands in separate text windows [4].

- What is the "reverse step" in GDB and how to enable it. Provide the key steps and commands.
  When a program is executed in reverse, the instructions that have most recently been executed are "un-executed", in reverse order. The program counter runs backward, following the previous thread of execution in reverse. As each instruction is "un-executed", the values of memory and/or registers that were changed by that instruction are reverted to their previous states. After executing a piece of source code in reverse, all side effects of that code should be "undone", and all variables should be returned to their prior values [6].
  `reverse-stepi [`*`count`*`]`: Reverse-execute one machine instruction.
  `reverse-continue [`*`ignore-count`*`]`: (also `rc [`*`ignore-count`*`]`) Beginning at the point where your program last stopped, start executing in reverse. Reverse execution will stop for breakpoints and synchronous exceptions (signals), just like normal execution.

```
(gdb) p $pc
(gdb) x 0x5e4c5d00
(gdb) bt
(gdb) reverse-stepi
(gdb) bt
(gdb) disas
```

```
7  (gdb) print $sp
8  (gdb) print *(long **) 0x7fffffffdc98
9  (gdb) watch *(long **) 0x7fffffffdc98
10 (gdb) reverse-continue
```

Line 4 and Line 10 are key steps and commands.

References

[1] Cody Arsenault. *The Pros and Cons of 8 Popular Databases*. KeyCDN, Apr. 20, 2017. URL: https://www.keycdn.com/blog/popular-databases.

[2] Jorge Campos. *Select last 7 days from unix timestamp mysql*. May 5, 2017. URL: https://stackoverflow.com/questions/42605114/select-last-7-days-from-unix-timestamp-mysql.

[3] Nick Fortescue. *How to find the average time difference between rows in a table?* May 18, 2009. URL: https://stackoverflow.com/questions/876842/how-to-find-the-average-time-difference-between-rows-in-a-table.

[4] GNU. *25 GDB Text User Interface*. URL: https://sourceware.org/gdb/onlinedocs/gdb/TUI.html.

[5] GNU. *3.10 Options for Debugging Your Program*. URL: https://gcc.gnu.org/onlinedocs/gcc/Debugging-Options.html.

[6] GNU. *6 Running programs backward*. URL: https://sourceware.org/gdb/onlinedocs/gdb/Reverse-Execution.html.

[7] GNU. *8.2 Backtraces*. URL: https://sourceware.org/gdb/current/onlinedocs/gdb/Backtrace.html#Backtrace.

[8] GNU. *GDB Documentation*. Version 12.0.50.20211017-git. Aug. 23, 2017. URL: https://sourceware.org/gdb/current/onlinedocs/gdb.pdf.

[9] GNU. *GDB: The GNU Project Debugger*. Oct. 8, 2021. URL: https://www.gnu.org/software/gdb/.