

Mel (<Chalk> Мел) Editor – MiniEditor for Linux.



Igor Lukyanov

February, 2025

Art2Dec

Introduction.

Mel (Mini Embedded Light Editor for Linux) is a terminal-based text editor designed for Linux and Unix platforms.

Written in C from scratch, Mel is highly minimalistic, dependency-independent, optimized for environments where resources are constrained.

Syntax highlighting for several programming languages and support for various platforms is already provided.

Mel aims to provide a lightweight yet powerful solution for developers working across diverse environments.

Main Goals

- To create a lightweight, minimalistic multiplatform text editor with no dependencies.
- To support resource-intensive environments such as Kubernetes pods, Docker containers, and IoT devices.
 - To provide syntax highlighting for multiple programming languages.
- To ensure compatibility with ARM64 architectures and other CPU types (Intel, AMD, etc.) and different Oses.

Key features.

- Multi-platform Support:
- Works seamlessly on Ubuntu Linux, Raspberry Pi OS, and macOS and it was tested on them.
- Supports various architectures, including ARM64 and x86_64, Apple M1, etc.
- Keybindings.
- Efficient keyboard shortcuts for navigation, editing, and functionality (refer to CLI help).
- Minimalistic Design.
- Dependency-free (does not rely on libraries like curses).

MacOS platform.

The screenshot shows a macOS desktop environment with two terminal windows open. The left terminal window displays the source code of the `mel.c` file, which is a command-line tool for highlighting code snippets. The right terminal window shows the help documentation for the `mel` command, detailing keybindings and options. The desktop dock at the bottom contains icons for various applications like Mail, Safari, and Finder. The system tray in the top right corner shows the date and time as "Sat Feb 1 2:28AM".

```
1 /*  
2 *  
3 *  
4 * This program is free software: you can redistribute it and/or modify  
5 * it under the terms of the GNU General Public License as published by  
6 * the Free Software Foundation, either version 3 of the License, or  
7 * any later version.  
8 *  
9 * This program is distributed in the hope that it will be useful,  
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of  
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
12 * GNU General Public License for more details.  
13 *  
14 * You should have received a copy of the GNU General Public License  
15 * along with this program. If not, see <http://www.gnu.org/licenses/>.  
16 */  
17  
18 /*** Include section ***/  
19  
20 // We add them above our includes, because the header  
21 // files we are including use the macros to decide what  
22 // features to expose. These macros remove some compilation  
23 // warnings. See  
24 // https://www.gnu.org/software/libc/manual/html\_node/Feature-Test-Macros.html  
25 // for more info.  
26 #define _DEFAULT_SOURCE  
27 #define _BSD_SOURCE  
28 #define _GNU_SOURCE  
29  
30 #include <ctype.h>  
31 #include <errno.h>  
32 #include <fcntl.h>  
33 #include <signal.h>  
34 #include <stdio.h>  
35 #include <stdarg.h>  
36 #include <stdlib.h>  
37 #include <stdbool.h>  
38 #include <string.h>  
39 #include <sys/types.h>  
40 #include <sys/stat.h>  
41 #include <sys/statfs.h>  
42 #include <terminio.h>  
43 #include <time.h>  
44 #include <unistd.h>  
45 #include <curl/curl.h>  
46 #include <json-c/json.h>  
47 #include <limits.h>  
48  
49 /*** Define section ***/  
50  
51 // This mimics the Ctrl + whatever behavior, setting the  
52 // 3 upper bits of the character pressed to 0.  
53 #define CTRL_KEY(k) ((k) & 0x1f)  
54 // Empty buffer  
55 #define ABUF_INIT {NULL, 0}  
56 // Version code  
57 #define MEL_VERSION "0.2.0"  
mel.c - 2873 lines
```

```
igorlukyanov@mbpro ~ % mel -h  
Usage: mel [OPTIONS] [FILE]  
  
KEYBINDINGS  
-----  
Keybinding Action  
Ctrl-Q Exit, 3 times click Ctrl-Q if file was changed without saving  
Ctrl-S Save, requires input of file name, if file didn't exist  
Ctrl-F Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching  
Ctrl-N Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only  
Ctrl-R Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only  
Ctrl-J Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input  
Ctrl-G Go to line Number, requires input the line number  
Ctrl-B Hide/Show line numbering  
Ctrl-E Flip line upwards  
Ctrl-D Flip line downwards  
Ctrl-C Copy line  
Ctrl-X Cut line  
Ctrl-V Paste line  
Ctrl-Z Undo  
Ctrl-Y Redo  
Ctrl-P Pause mel (type "fg" to resume)  
Ctrl-W Retrieve Ollama LLM response  
Ctrl-H Toggle this help screen  
Home Move the cursor to the beginning of the line  
End Move cursor to end of line  
PgUp Up page scroll  
PgDn Down page scroll  
Up Move cursor up one position  
Down Move cursor down one position  
Left Move cursor left one position  
Right Move cursor right one position  
Backspace Delete character  
  
OPTIONS  
-----  
Option Action  
-h | --help Prints the help  
-v | --version Prints the version of mel  
-b | --backup Create backup (.bak) file before saving  
-l | --line <number> <file_name> Open file with cursor on specified line number  
-w | --width <columns> Set visual column width marker  
  
Supports highlighting for C,C++,Java,Bash,Mshell,Python,PHP,Javascript,JSON,XML,SQL,Ruby,Go  
License: Public domain libre software GPL3,v.0.2.0, 2025  
Initial coding: Igor Lukyanov, igor.lukyanov@appservgrid.com  
For now, usage of UTF-8 is recommended.  
igorlukyanov@mbpro ~ %
```

Art2Dec

Raspberry PI OS.

The screenshot shows a terminal window on a Raspberry Pi OS desktop environment. The terminal title is "oracle@raspberrypi: ~". The window displays the usage of the "mel" command and its keybindings.

```
oracle@raspberrypi:~$ mel -h
Usage: mel [OPTIONS] [FILE]

File Edit Tabs Help
```

KEYBINDINGS

Keybinding	Action
Ctrl-Q	Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S	Save, requires input of file name, if file didn't exist
Ctrl-F	Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching
Ctrl-N	Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R	Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-J	Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input
Ctrl-G	Go to line Number, requires input the line number
Ctrl-B	Hide/Show line numbering
Ctrl-U	Flip line upwards
Ctrl-D	Flip line downwards
Ctrl-C	Copy line
Ctrl-X	Cut line
Ctrl-V	Paste line
Ctrl-Z	Undo
Ctrl-Y	Redo
Ctrl-P	Pause mel (type "fg" to resume)
Ctrl-W	Retrieve Olama LLM response
Ctrl-H	Toggle this help screen
Home	Move the cursor to the beginning of the line
End	Move cursor to end of line
PgUp	Up page scroll
PgDn	Down page scroll
Up	Move cursor up one position
Down	Move cursor down one position
Left	Move cursor left one position
Right	Move cursor right one position
Backspace	Delete character

OPTIONS

Option	Action
-h --help	Prints the help
-v --version	Prints the version of mel
-b --backup	Create backup (.bak) file before saving
-l --line <number> <file_name>	Open file with cursor on specified line number
-w --width <columns>	Set visual column width marker

Supports highlighting for C,C++,Java,Bash,Mshell,Python,PHP,Javascript,JSON,XML,SQL,Ruby,Go
License: Public domain libre software GPL3,v.0.2.0, 2025
Initial coding: Igor Lukyanov, igor.lukyanov@appservgrid.com

Line: For now, usage of UTF-8 is recommended.
oracle@raspberrypi:~\$

Art2Dec

Ubuntu Linux 2x.0x OS and other Linux like Debian, Linux Mint, CentOS, Fedora, Oracle Linux, RedHat EL, etc.

Activities Terminal Feb 1 02:45 oracle@asus: ~

Home

oracle@asus: /home/igor/mshell

```
1150     char clean_query[MAX_COMMAND_LEN];
1151     strcpy(clean_query, query, sizeof(clean_query) - 1);
1152     clean_query[sizeof(clean_query) - 1] = '\0';
1153     size_t len = strlen(clean_query);
1154     if (len > 0 && clean_query[len-1] == '') {
1155         clean_query[len-1] = '\0';
1156     }
1157
1158     int stdout_copy = -1;
1159     if (is_pipe || is_tee) {
1160         stdout_copy = dup(STDOUT_FILENO);
1161         dup2(STDOUT_FILENO, STDERR_FILENO);
1162     }
1163
1164     call_ollama(clean_query, 1, 1);
1165
1166     if (is_pipe || is_tee) {
1167         dup2(stdout_copy, STDOUT_FILENO);
1168         close(stdout_copy);
1169     }
1170     return;
1171 }
1172 else if (strncmp(expanded_command, "ollama2 ", 8) == 0) {
1173     const char *query = expanded_command + 8;
1174     bool is_pipe = strstr(query, " --pipe ") || strstr(query, " -p ");
1175     bool is_tee = strstr(query, " --tee ") || strstr(query, " -t ");
1176
1177     char clean_query[MAX_COMMAND_LEN];
1178     strcpy(clean_query, query, sizeof(clean_query) - 1);
1179     clean_query[sizeof(clean_query) - 1] = '\0';
1180
1181     int stdout_copy = -1;
1182     if (is_pipe || is_tee) {
1183         stdout_copy = dup(STDOUT_FILENO);
1184         dup2(STDOUT_FILENO, STDERR_FILENO);
1185     }
1186
1187     call_ollama(clean_query, 2, 0);
1188
1189     if (is_pipe || is_tee) {
1190         dup2(stdout_copy, STDOUT_FILENO);
1191         close(stdout_copy);
1192     }
1193     return;
1194 }
1195
1196
1197
1198
1199
1200
```

KEYBINDINGS

Keybinding	Action
Ctrl-Q	Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S	Save, requires input of file name, if file didn't exist
Ctrl-F	Search by pattern, Esc - exit from Search, works after Ctrl-F only
Ctrl-N	Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R	Backward Search by pattern after Ctrl-F. Esc - exit from Search, Enter and Arrows to interact
Ctrl-J	Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to
input	Go to line Number, requires input the line number
Ctrl-G	Hide/Show line numbering
Ctrl-B	Flip line upwards
Ctrl-E	Flip line downwards
Ctrl-D	Copy line
Ctrl-C	Cut line
Ctrl-X	Paste line
Ctrl-V	Undo
Ctrl-Z	Redo
Ctrl-Y	Pause mel (type "fg" to resume)
Ctrl-P	Retrieve Ollama LLM response
Ctrl-H	Toggle this help screen
Home	Move the cursor to the beginning of the line
End	Move cursor to end of line
PgUp	Up page scroll
PgDn	Down page scroll
Up	Move cursor up one position
Down	Move cursor down one position
Left	Move cursor left one position
Right	Move cursor right one position
Backspace	Delete character

OPTIONS

Option	Action
-h --help	Prints the help
-v --version	Prints the version of mel
-b --backup	Create backup (.bak) file before saving
-l --line <number> <file_name>	Open file with cursor on specified line number
-w --width <columns>	Set visual column width marker

Supports highlighting for C,C++,Java,Bash,Mshell,Python,PHP,Javascript,JSON,XML,SQL,Ruby,Go.
License: Public domain libre software GPL3,v.0.2.0, 2025
Initial coding: Igor Lukyanov, igor.lukyanov@appservgrid.com
For now, usage of UTF-8 is recommended.

Press any key to continue... Line 1200/1581 Col 1

mshell.c - 1581 lines

Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)

CLI help screen with Keybindings and start Options

```
oracle@asus: ~/Desktop
Usage: mel [OPTIONS] [FILE]

KEYBINDINGS
-----
Keybinding      Action
Ctrl-Q          Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S          Save, requires input of file name, if file didn't exist
Ctrl-F          Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching
Ctrl-N          Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R          Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-J          Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input
Ctrl-G          Go to line Number, requires input the line number
Ctrl-B          Hide/Show line numbering
Ctrl-E          Flip line upwards
Ctrl-D          Flip line downwards
Ctrl-C          Copy line
Ctrl-X          Cut line
Ctrl-V          Paste line
Ctrl-Z          Undo
Ctrl-Y          Redo
Ctrl-P          Pause mel (type "fg" to resume)
Ctrl-W          Retrieve Ollama LLM response
Ctrl-H          Toggle this help screen
Home           Move the cursor to the beginning of the line
End             Move cursor to end of line
PgUp            Up page scroll
PgDn            Down page scroll
Up               Move cursor up one position
Down             Move cursor down one position
Left             Move cursor left one position
Right            Move cursor right one position
Backspace        Delete character

OPTIONS
-----
Option          Action
-h | --help      Prints the help
-v | --version   Prints the version of mel
-b | --backup    Create backup (.bak) file before saving
-l | --line <number> <file_name> Open file with cursor on specified line number
-w | --width <columns> Set visual column width marker
-----
Supports highlighting for C,C++,Java,Bash,Mshell,Python,PHP,Javascript,JSON,XML,SQL,Ruby,Go
License: Public domain libre software GPL3,v.0.2.0, 2025
Initial coding: Igor Lukyanov, igor.lukyanov@appservgrid.com
For now, usage of UTF-8 is recommended.
/home/igor >
```

Modern current supported version.

```
oracle@asus:~/Desktop$ mel --version
mel - version 0.2.0
oracle@asus:~/Desktop$
```

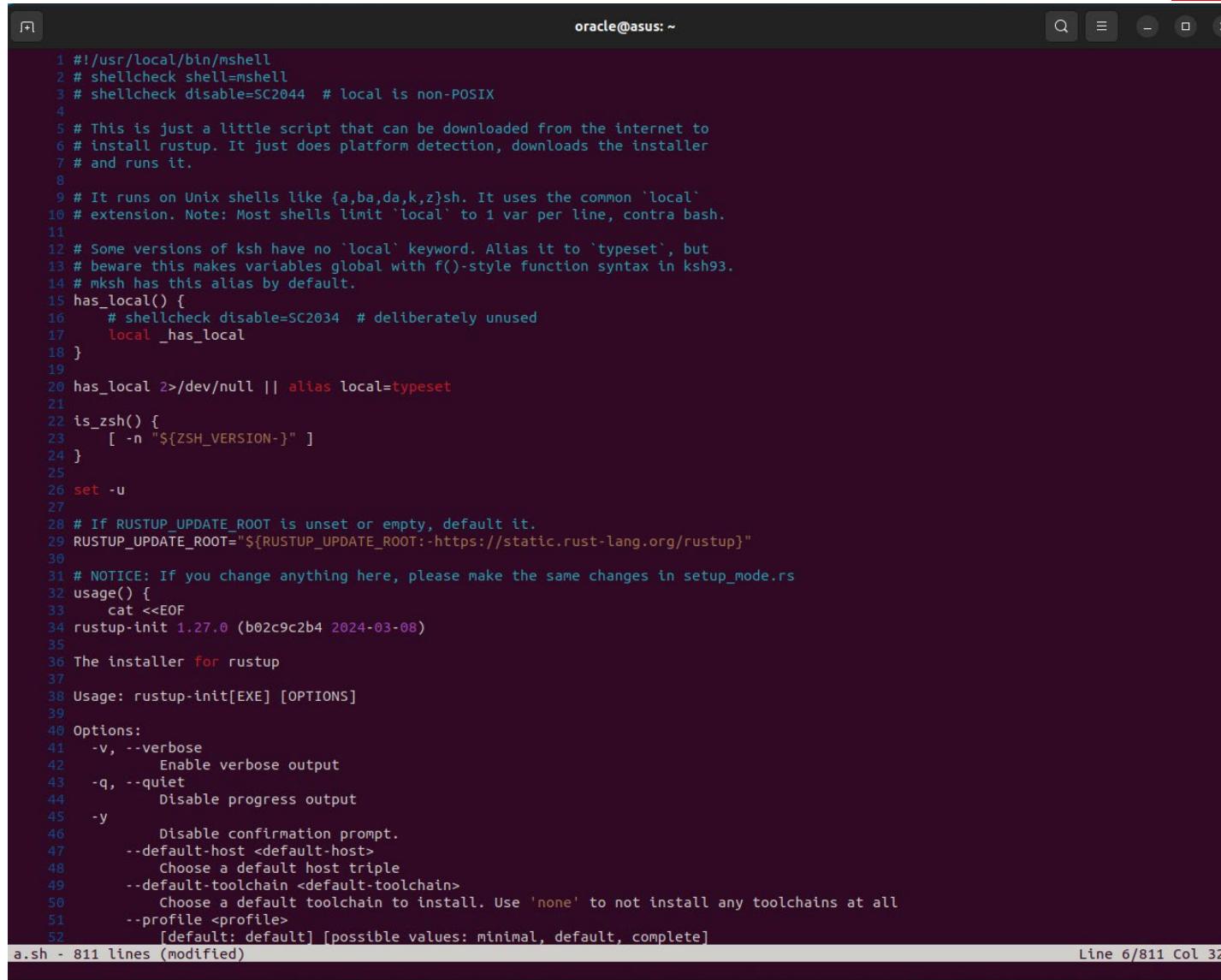
Highlighting Supported Languages.

- Mel supports syntax highlighting for a wide range of programming languages:
- C, C++, Java, Bash, Mshell, Python, PHP
- JavaScript, JSON, XML, SQL, Ruby, Go
- Etc. It is also possible to add data syntax format for highlighting by your request.

Main data formats for syntax highlighting.

Name	Date modified	Type	Size
alternative_characters.rb	1/30/2025 4:07 AM	Ruby Source File	1 KB
Candidate.sql	1/30/2025 4:02 AM	SQL Source File	1 KB
database1.c	1/22/2025 11:36 PM	C Source File	7 KB
date.cpp	1/30/2025 2:31 AM	C++ Source File	1 KB
date.h	1/30/2025 2:29 AM	C Header File	1 KB
even_or_odd.js	1/30/2025 3:34 AM	JavaScript File	1 KB
examples.phtml	1/30/2025 3:49 AM	PHP HTML Source...	1 KB
glossary.json	1/30/2025 3:51 AM	JSON File	1 KB
lambdas.pyo	1/30/2025 3:14 AM	PYO File	1 KB
logResults.jsonp	1/30/2025 3:56 AM	JSONP File	1 KB
Main.java	1/30/2025 2:47 AM	IntelliJ IDEA	1 KB
math_part124.ms	1/28/2025 1:18 AM	MS File	3 KB
maxnumber.py	1/30/2025 2:59 AM	JetBrains PyCharm	1 KB
my_module.pyc	1/30/2025 3:09 AM	PYC File	1 KB
MyComponent.jsx	1/30/2025 3:45 AM	JSX File	1 KB
person.hpp	1/30/2025 2:36 AM	C++ Header File	1 KB
readlines.sh	1/30/2025 3:19 AM	Shell Script	1 KB
sort_test.go	1/30/2025 4:10 AM	Go Source File	1 KB
tesla.php	1/30/2025 3:44 AM	PHP Source File	1 KB
testhi.cc	1/30/2025 2:43 AM	C++ Source	1 KB
textui.pyw	1/30/2025 2:54 AM	Python source file	12 KB
toa.py3	1/30/2025 3:06 AM	PY3 File	1 KB
widget.xml	1/30/2025 3:58 AM	Microsoft Edge H...	1 KB

Mshell highlighting support.

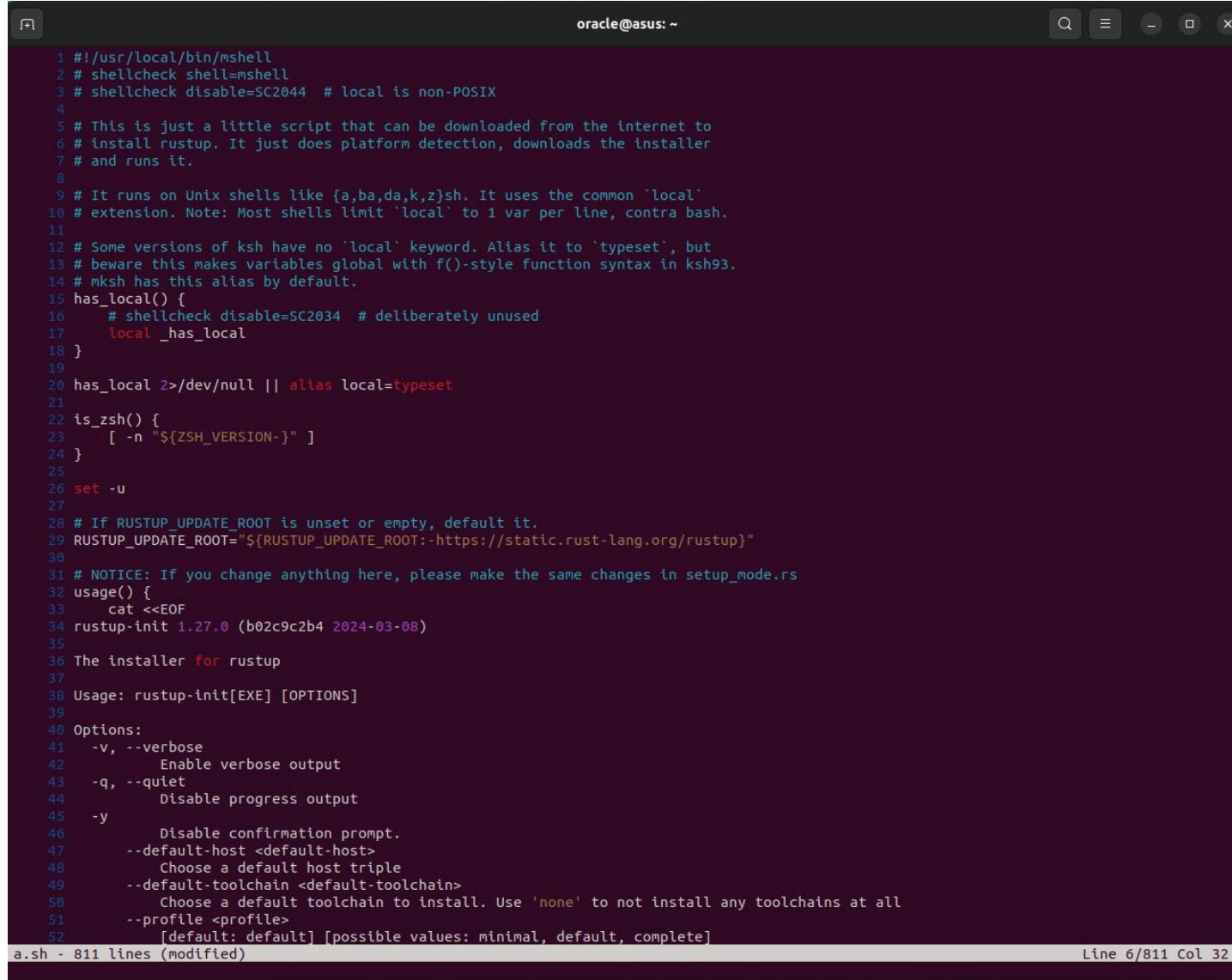


A screenshot of a terminal window titled "oracle@asus: ~". The window displays a script file with syntax highlighting. The script is a shell script for rustup-init, written in Mshell. The code uses various colors to highlight different parts of the script, such as blue for command names like "#!", "#", "alias", "set", "rustup-init", and "cat"; red for variables like "\$RUSTUP_UPDATE_ROOT"; green for comments; and orange for function definitions like "has_local()". The terminal interface includes a title bar, a menu bar with icons for search, list, and close, and a status bar at the bottom indicating "Line 6/811 Col 32".

```
1 #!/usr/local/bin/mshell
2 # shellcheck shell=mshell
3 # shellcheck disable=SC2044 # local is non-POSIX
4
5 # This is just a little script that can be downloaded from the internet to
6 # install rustup. It just does platform detection, downloads the installer
7 # and runs it.
8
9 # It runs on Unix shells like {a,ba,da,k,z}sh. It uses the common 'local'
10 # extension. Note: Most shells limit 'local' to 1 var per line, contra bash.
11
12 # Some versions of ksh have no 'local' keyword. Alias it to 'typeset', but
13 # beware this makes variables global with f()-style function syntax in ksh93.
14 # mksh has this alias by default.
15 has_local() {
16     # shellcheck disable=SC2034 # deliberately unused
17     local _has_local
18 }
19
20 has_local 2>/dev/null || alias local=typeset
21
22 is_zsh() {
23     [ -n "${ZSH_VERSION-}" ]
24 }
25
26 set -u
27
28 # If RUSTUP_UPDATE_ROOT is unset or empty, default it.
29 RUSTUP_UPDATE_ROOT="${RUSTUP_UPDATE_ROOT:-https://static.rust-lang.org/rustup}"
30
31 # NOTICE: If you change anything here, please make the same changes in setup_mode.rs
32 usage() {
33     cat <<EOF
34 rustup-init 1.27.0 (b02c9c2b4 2024-03-08)
35
36 The installer for rustup
37
38 Usage: rustup-init[EXE] [OPTIONS]
39
40 Options:
41     -v, --verbose
42             Enable verbose output
43     -q, --quiet
44             Disable progress output
45     -y
46             Disable confirmation prompt.
47     --default-host <default-host>
48             Choose a default host triple
49     --default-toolchain <default-toolchain>
50             Choose a default toolchain to install. Use 'none' to not install any toolchains at all
51     --profile <profile>
52             [default: default] [possible values: minimal, default, complete]
```

a.sh - 811 lines (modified) Line 6/811 Col 32

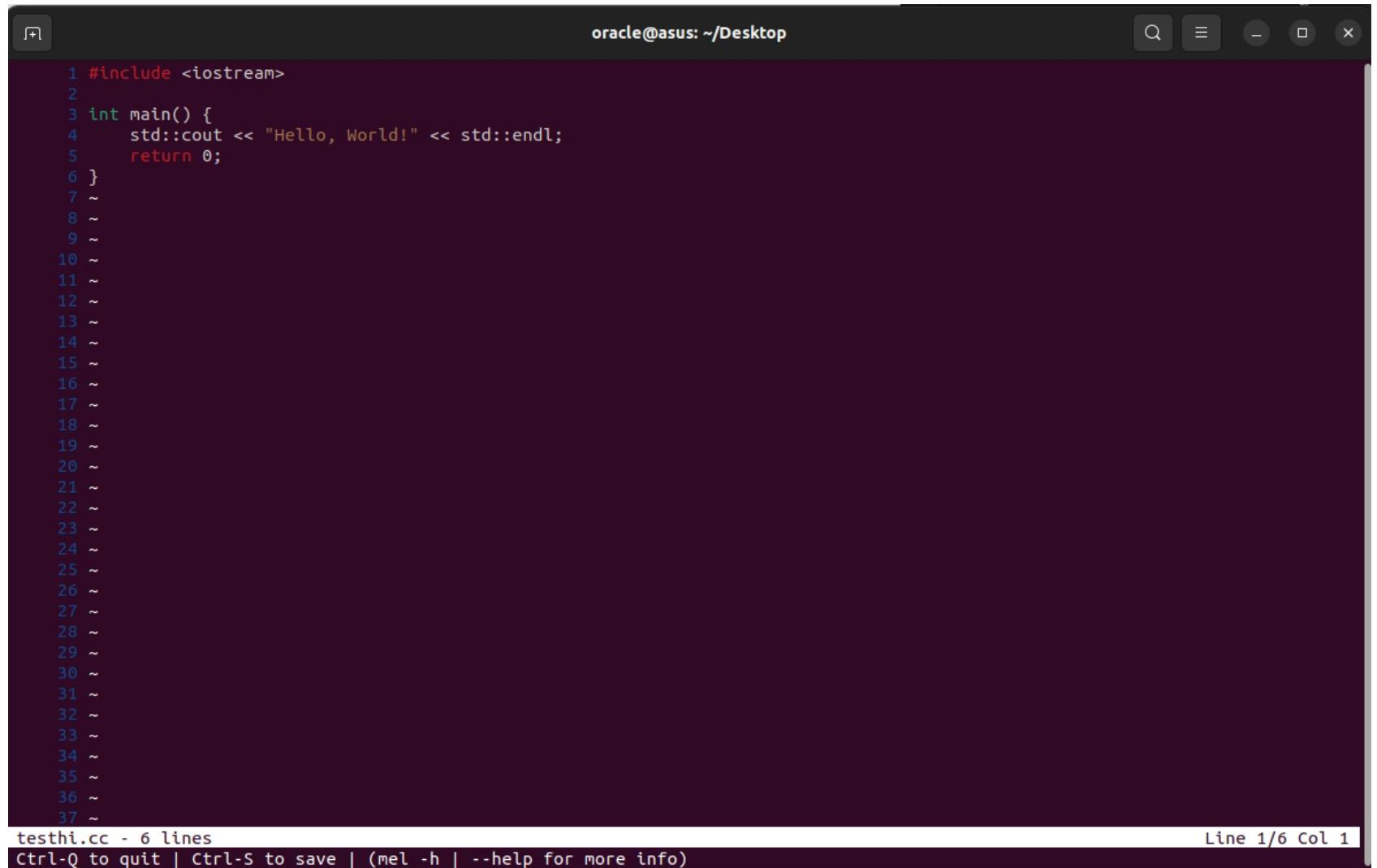
Bash highlighting support.



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "oracle@asus: ~". The window contains a script with line numbers from 1 to 52. The script uses several shell constructs like `#!/usr/local/bin/mshell`, `# shellcheck shell=mshell`, and `# shellcheck disable=SC2044`. It includes comments explaining its purpose for non-POSIX shells and handles different shells (ksh, zsh) by aliasing `local` to `typeset` or using `mksh`'s built-in alias. It also sets up `RUSTUP_UPDATE_ROOT` and provides usage and option information for `rustup-init`. The bottom status bar shows "a.sh - 811 lines (modified)" and "Line 6/811 Col 32".

```
1 #!/usr/local/bin/mshell
2 # shellcheck shell=mshell
3 # shellcheck disable=SC2044 # local is non-POSIX
4
5 # This is just a little script that can be downloaded from the internet to
6 # install rustup. It just does platform detection, downloads the installer
7 # and runs it.
8
9 # It runs on Unix shells like {a,ba,da,k,z}sh. It uses the common 'local'
10 # extension. Note: Most shells limit 'local' to 1 var per line, contra bash.
11
12 # Some versions of ksh have no 'local' keyword. Alias it to 'typeset', but
13 # beware this makes variables global with f()-style function syntax in ksh93.
14 # mksh has this alias by default.
15 has_local() {
16     # shellcheck disable=SC2034 # deliberately unused
17     local _has_local
18 }
19
20 has_local 2>/dev/null || alias local=typeset
21
22 is_zsh() {
23     [ -n "${ZSH_VERSION-}" ]
24 }
25
26 set -u
27
28 # If RUSTUP_UPDATE_ROOT is unset or empty, default it.
29 RUSTUP_UPDATE_ROOT="${RUSTUP_UPDATE_ROOT:-https://static.rust-lang.org/rustup}"
30
31 # NOTICE: If you change anything here, please make the same changes in setup_mode.rs
32 usage() {
33     cat <<EOF
34 rustup-init 1.27.0 (b02c9c2b4 2024-03-08)
35
36 The installer for rustup
37
38 Usage: rustup-init[EXE] [OPTIONS]
39
40 Options:
41     -v, --verbose
42             Enable verbose output
43     -q, --quiet
44             Disable progress output
45     -y
46             Disable confirmation prompt.
47     --default-host <default-host>
48             Choose a default host triple
49     --default-toolchain <default-toolchain>
50             Choose a default toolchain to install. Use 'none' to not install any toolchains at all
51     --profile <profile>
52             [default: default] [possible values: minimal, default, complete]
```

C++_cc highlighting support.

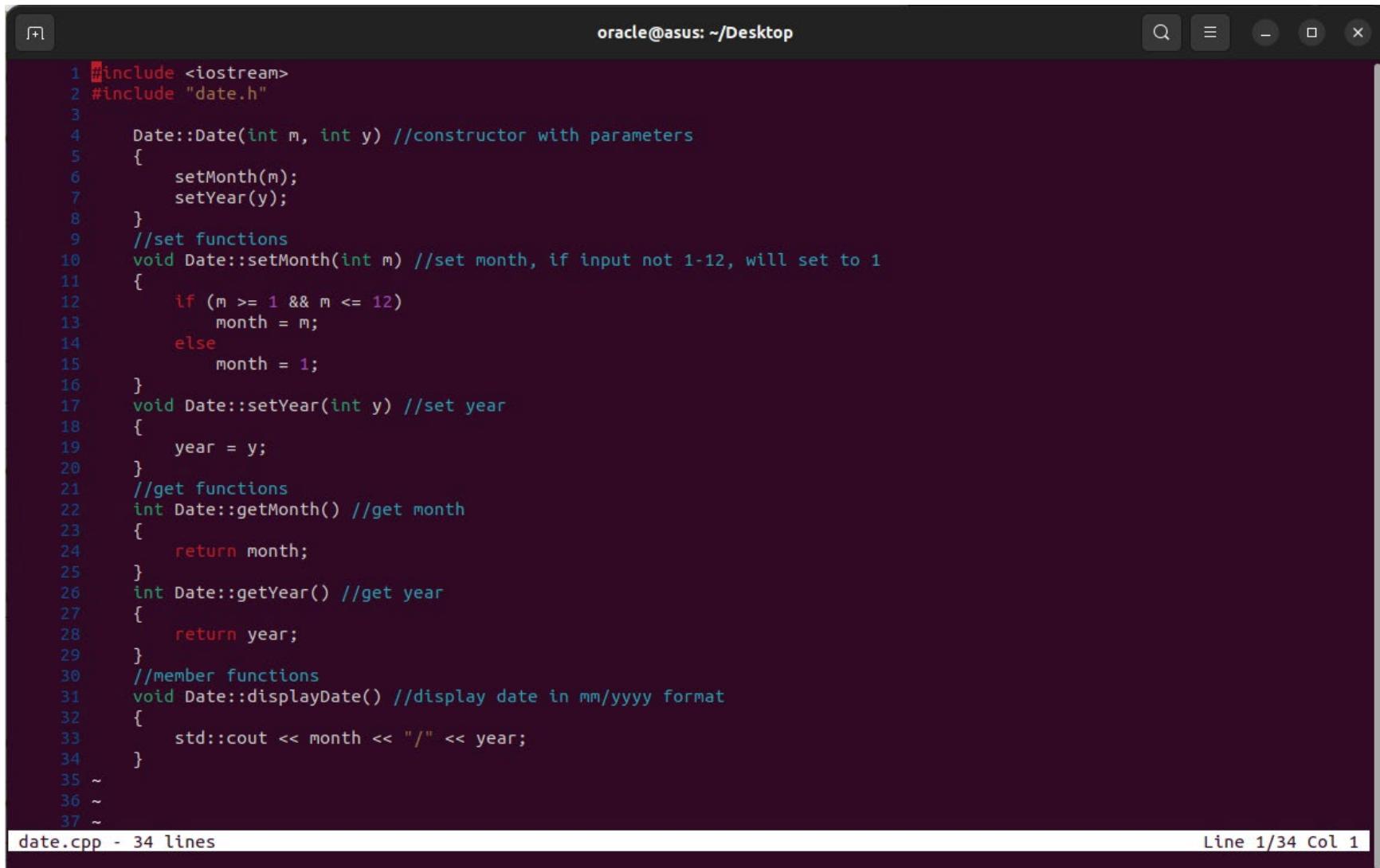


A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains a C++ program named "testhi.cc" with line numbers 1 through 37. The code is as follows:

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello, World!" << std::endl;
5     return 0;
6 }
7 ~
8 ~
9 ~
10 ~
11 ~
12 ~
13 ~
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
36 ~
37 ~
```

The terminal also displays the message "testhi.cc - 6 lines" and the status "Line 1/6 Col 1". A status bar at the bottom shows "Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)".

C++_cpp highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window displays a C++ program named "date.cpp" with line numbers from 1 to 37. The code includes #include directives for `<iostream>` and `"date.h"`, a constructor `Date::Date(int m, int y)`, and member functions `setMonth`, `setYear`, `getMonth`, `getYear`, and `displayDate`. The code is color-coded: numbers are blue, strings are purple, and keywords like `#include`, `int`, `void`, `if`, `else`, `return`, and `std::cout` are red. The terminal interface includes a search bar, a menu icon, and standard window control buttons.

```
1 #include <iostream>
2 #include "date.h"
3
4     Date::Date(int m, int y) //constructor with parameters
5     {
6         setMonth(m);
7         setYear(y);
8     }
9     //set functions
10    void Date::setMonth(int m) //set month, if input not 1-12, will set to 1
11    {
12        if (m >= 1 && m <= 12)
13            month = m;
14        else
15            month = 1;
16    }
17    void Date::setYear(int y) //set year
18    {
19        year = y;
20    }
21    //get functions
22    int Date::getMonth() //get month
23    {
24        return month;
25    }
26    int Date::getYear() //get year
27    {
28        return year;
29    }
30    //member functions
31    void Date::displayDate() //display date in mm/yyyy format
32    {
33        std::cout << month << "/" << year;
34    }
35 ~
36 ~
37 ~
```

date.cpp - 34 lines

Line 1/34 Col 1

C++_hpp header highlighting support.

```
oracle@asus: ~/Desktop
```

```
1 #ifndef PERSON_HPP
2 #define PERSON_HPP
3
4 #include <string>
5
6 class Person {
7 private:
8     std::string name;
9     int age;
10
11 public:
12     Person();
13     Person(const std::string& name, int age);
14     void setName(const std::string& newName);
15     void setAge(int newAge);
16     std::string getName() const;
17     int getAge() const;
18     void printInfo() const;
19 };
20
21 #endif
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
36 ~
37 ~
```

person.hpp - 21 lines

ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)

Line 1/21 Col 1

C++ header highlighting support.

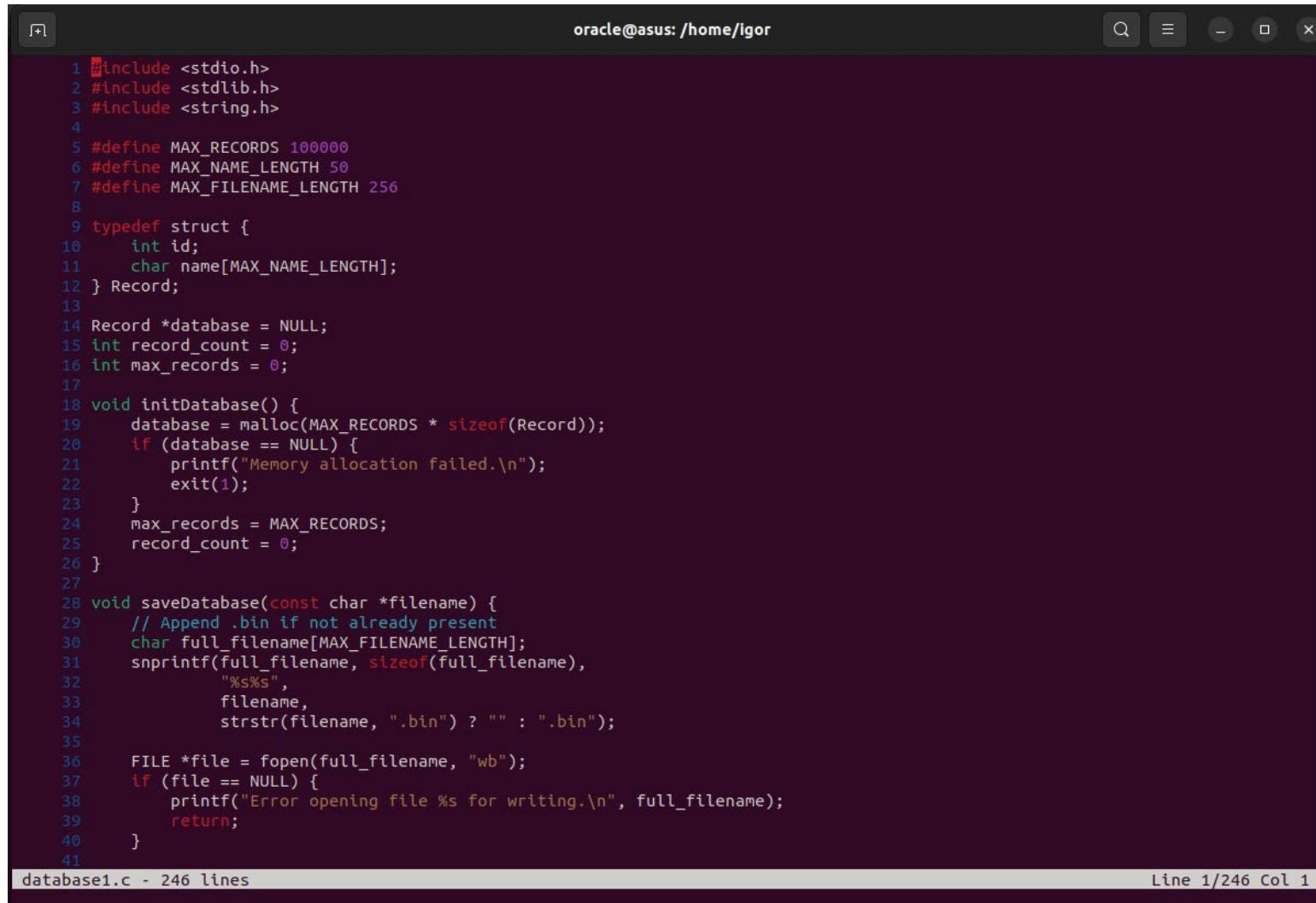
A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window displays a C++ header file named "date.h" with line numbers 1 through 31. The code defines a Date class with public and private members, including constructors, set and get functions, and a member function for displaying the date. The code is color-coded for syntax highlighting, and the terminal interface includes standard window controls at the top right.

```
1 #ifndef DATE_H
2 #define DATE_H
3
4 class Date
5 {
6     public:
7
8     //constructor with default values if not provided
9     Date(int = 1, int = 2000); ■
10
11    //set functions
12    void setMonth(int); //set month
13    void setYear(int); //set year
14
15    //get functions
16    int getMonth(); //get month
17    int getYear(); //get year
18
19    //member functions
20    void displayDate(); //display date in mm/dd/yyyy format
21
22    private:
23
24    int month;
25    int year;
26 };
27
28 #endif
29 ~
30 ~
31 ~
```

date.h - 28 lines

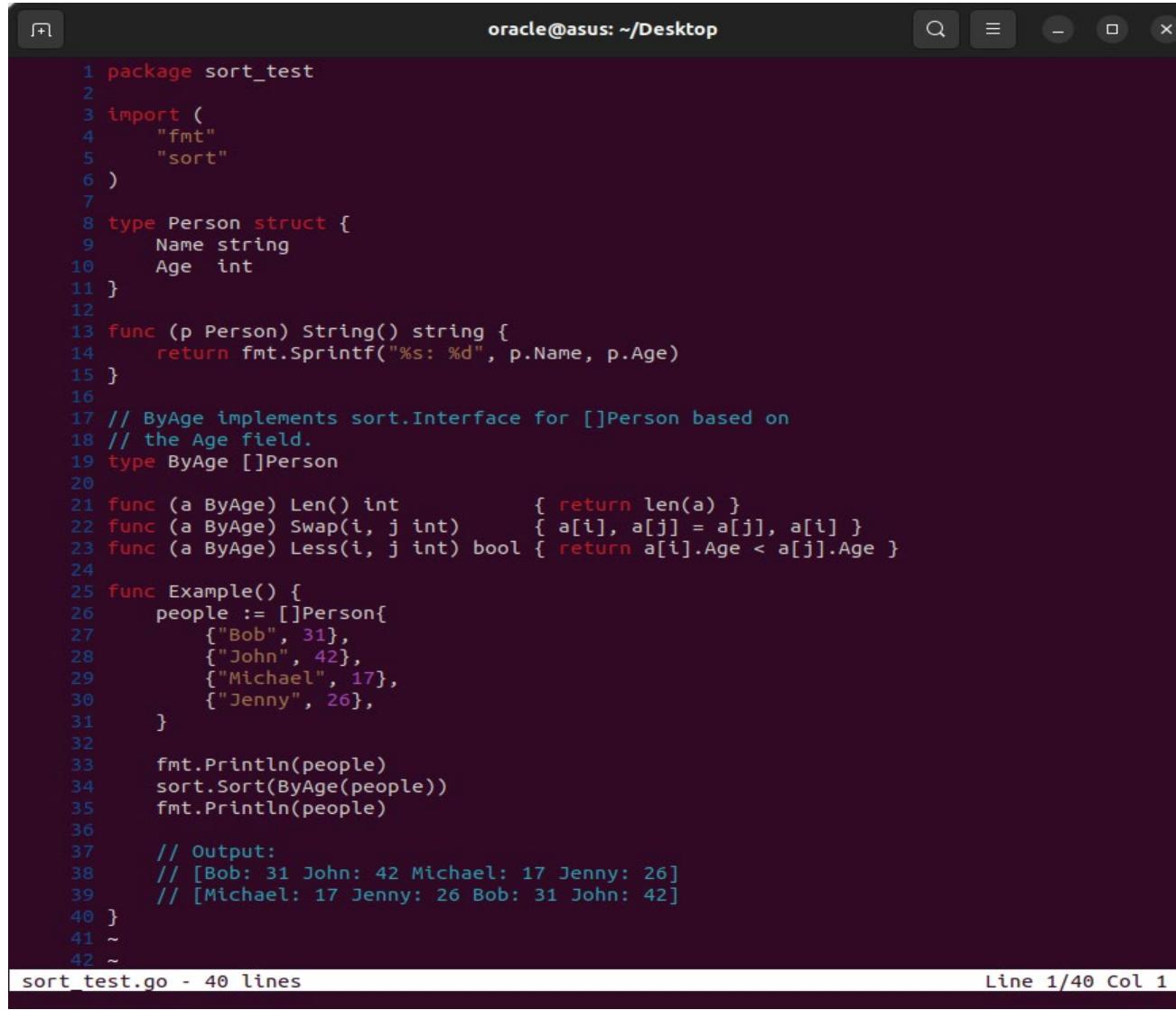
Line 9/28 Col 29

C lang highlighting support (ANSI C, C89 (ANSI X3.159-1989), ISO C, C90 (ISO/IEC 9899:1990), C99, C9X (ISO/IEC 9899:1999), C11, C1X (ISO/IEC 9899:2011), C17, C18 (ISO/IEC 9899:2018), C23, C2X (ISO/IEC 9899:2024}).



```
oracle@asus: /home/igor
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define MAX_RECORDS 100000
6 #define MAX_NAME_LENGTH 50
7 #define MAX_FILENAME_LENGTH 256
8
9 typedef struct {
10     int id;
11     char name[MAX_NAME_LENGTH];
12 } Record;
13
14 Record *database = NULL;
15 int record_count = 0;
16 int max_records = 0;
17
18 void initDatabase() {
19     database = malloc(MAX_RECORDS * sizeof(Record));
20     if (database == NULL) {
21         printf("Memory allocation failed.\n");
22         exit(1);
23     }
24     max_records = MAX_RECORDS;
25     record_count = 0;
26 }
27
28 void saveDatabase(const char *filename) {
29     // Append .bin if not already present
30     char full_filename[MAX_FILENAME_LENGTH];
31     snprintf(full_filename, sizeof(full_filename),
32             "%s%s",
33             filename,
34             strstr(filename, ".bin") ? "" : ".bin");
35
36     FILE *file = fopen(full_filename, "wb");
37     if (file == NULL) {
38         printf("Error opening file %s for writing.\n", full_filename);
39         return;
40     }
41 }
database1.c - 246 lines
Line 1/246 Col 1
```

Go language highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window displays a Go program named "sort_test.go". The code uses syntax highlighting to distinguish between different language constructs. The background of the terminal is dark, and the highlighted text is in various colors including red, blue, green, and yellow. The code defines a "Person" struct, a "ByAge" type, and implements the sort.Interface for []Person based on age. It includes an Example() function that prints a list of people and sorts them by age. The terminal status bar at the bottom shows "sort test.go - 40 lines" and "Line 1/40 Col 1".

```
1 package sort_test
2
3 import (
4     "fmt"
5     "sort"
6 )
7
8 type Person struct {
9     Name string
10    Age int
11 }
12
13 func (p Person) String() string {
14     return fmt.Sprintf("%s: %d", p.Name, p.Age)
15 }
16
17 // ByAge implements sort.Interface for []Person based on
18 // the Age field.
19 type ByAge []Person
20
21 func (a ByAge) Len() int           { return len(a) }
22 func (a ByAge) Swap(i, j int)     { a[i], a[j] = a[j], a[i] }
23 func (a ByAge) Less(i, j int) bool { return a[i].Age < a[j].Age }
24
25 func Example() {
26     people := []Person{
27         {"Bob", 31},
28         {"John", 42},
29         {"Michael", 17},
30         {"Jenny", 26},
31     }
32
33     fmt.Println(people)
34     sort.Sort(ByAge(people))
35     fmt.Println(people)
36
37     // Output:
38     // [Bob: 31 John: 42 Michael: 17 Jenny: 26]
39     // [Michael: 17 Jenny: 26 Bob: 31 John: 42]
40 }
41 ~
42 ~
```

sort test.go - 40 lines Line 1/40 Col 1

Java language highlighting support.

A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains Java code for reversing a number. The code is color-coded: class and method names are in red, while numbers and strings are in blue. The terminal shows line numbers from 1 to 37 on the left, and the status bar at the bottom indicates "Main.java - 21 lines" and "Line 1/21 Col 1".

```
1 class Main {  
2     public static void main(String[] args) {  
3         int num = 1234, reversed = 0;  
4         System.out.println("Original Number: " + num);  
5         // run loop until num becomes 0  
6         while(num != 0) {  
7             // get last digit from num  
8             int digit = num % 10;  
9             reversed = reversed * 10 + digit;  
10            // remove the last digit from num  
11            num /= 10;  
12        }  
13        System.out.println("Reversed Number: " + reversed);  
14    }  
15    ~  
16    ~  
17    ~  
18    ~  
19    ~  
20    ~  
21    ~  
22    ~  
23    ~  
24    ~  
25    ~  
26    ~  
27    ~  
28    ~  
29    ~  
30    ~  
31    ~  
32    ~  
33    ~  
34    ~  
35    ~  
36    ~  
37    ~  
Main.java - 21 lines  
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)
```

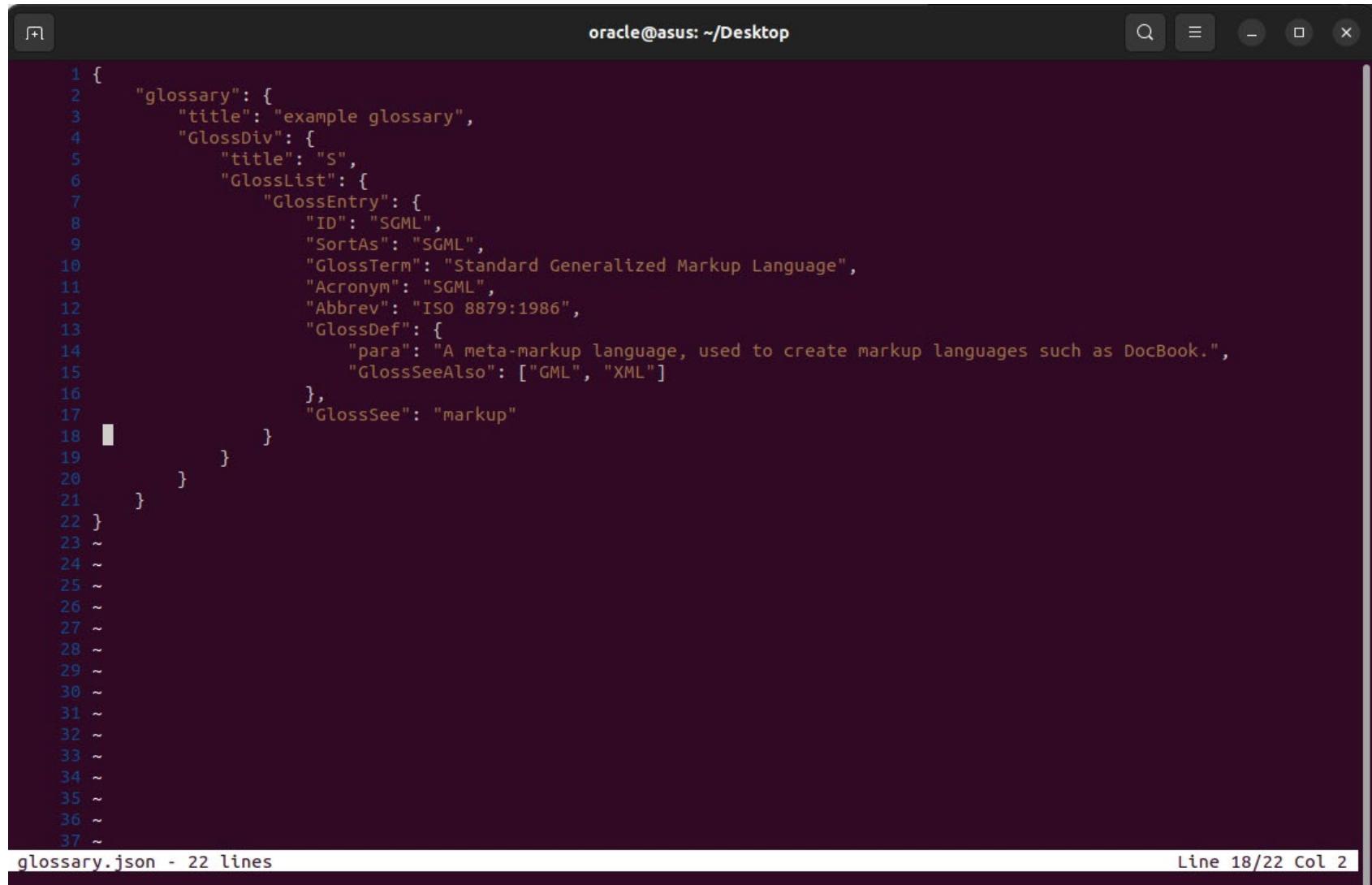
JavaScript language highlighting support.

A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains a code editor with the following JavaScript code:

```
1 // program to check if the number is even or odd
2 // take input from the user
3 const number = prompt("Enter a number: ");
4
5 //check if the number is even
6 if(number % 2 == 0) {
7     console.log("The number is even.");
8 }
9
10 // if the number is odd
11 else {
12     console.log("The number is odd.");
13 }
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
```

The code editor highlights syntax: numbers in blue, strings in orange, and keywords like "const" and "if" in red. The status bar at the bottom shows "even_or_odd.js - 13 lines" and "Line 1/13 Col 1".

JSON files formatting highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window displays a JSON file named "glossary.json" with 22 lines of code. The code is color-coded for syntax highlighting, with numbers on the left indicating line numbers. The JSON structure includes a root object with a "glossary" key, which contains a "title" key ("example glossary") and a "GlossDiv" key. The "GlossDiv" key has its own "title" key ("S") and a "GlossList" key. The "GlossList" key contains a single "GlossEntry" key, which has "ID" ("SGML"), "SortAs" ("SGML"), "GlossTerm" ("Standard Generalized Markup Language"), "Acronym" ("SGML"), "Abbrev" ("ISO 8879:1986"), and a "GlossDef" key containing a "para" describing SGML and a "GlossSeeAlso" array with ["GML", "XML"]. The "GlossDef" key also has a "GlossSee" key pointing to "markup". The terminal window has a dark background and standard Linux-style window controls at the top right.

```
1 {
2     "glossary": {
3         "title": "example glossary",
4         "GlossDiv": {
5             "title": "S",
6             "GlossList": {
7                 "GlossEntry": {
8                     "ID": "SGML",
9                     "SortAs": "SGML",
10                    "GlossTerm": "Standard Generalized Markup Language",
11                    "Acronym": "SGML",
12                    "Abbrev": "ISO 8879:1986",
13                    "GlossDef": {
14                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
15                        "GlossSeeAlso": ["GML", "XML"]
16                    },
17                    "GlossSee": "markup"
18                }
19            }
20        }
21    }
22 }
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
36 ~
37 ~
```

glossary.json - 22 lines Line 18/22 Col 2

JSONP files formatting highlighting support.

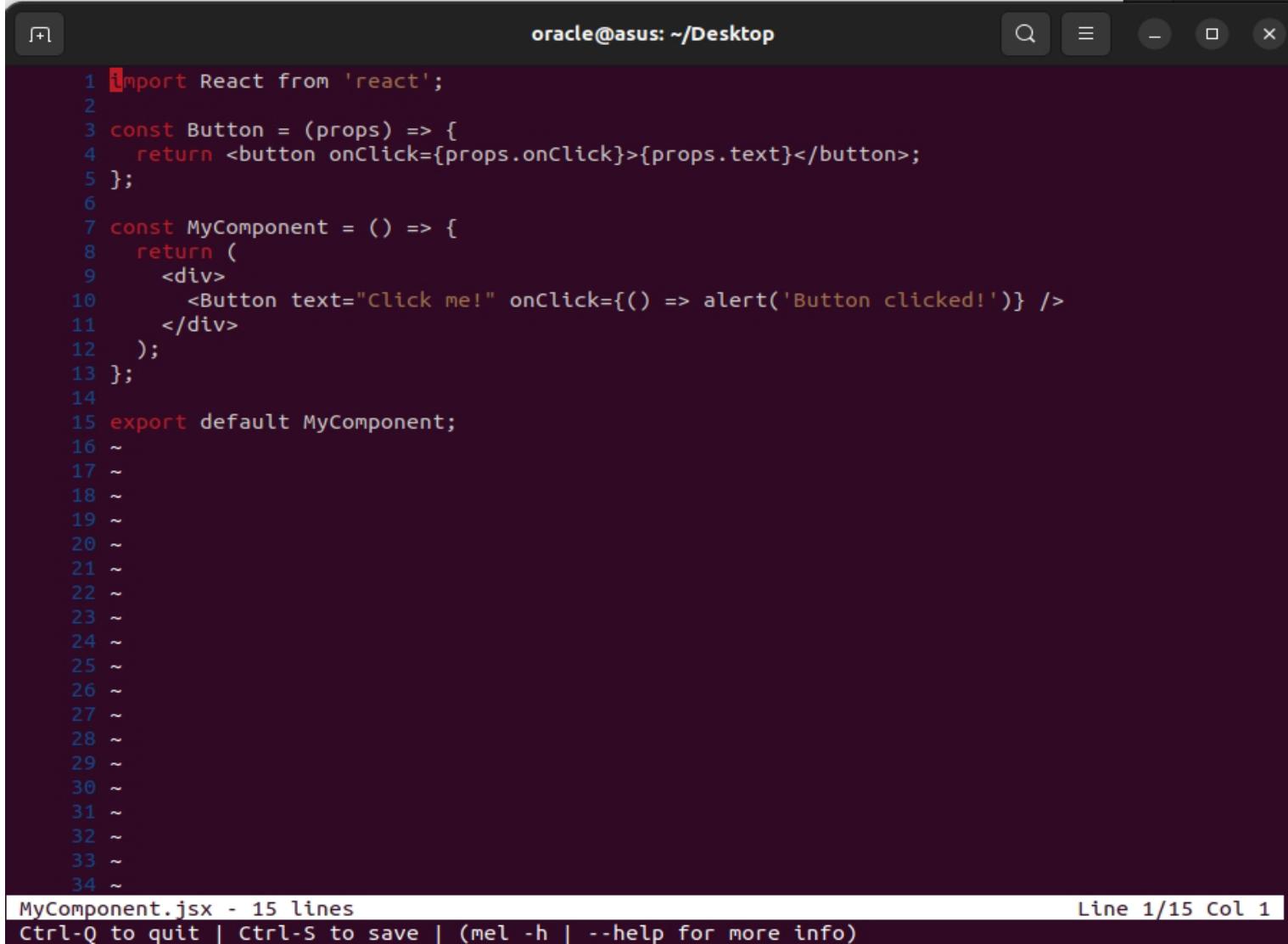


A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following code:

```
1 function logResults(json){  
2   console.log(json);  
3 }  
4  
5 $.ajax({  
6   url: "https://api.github.com/users/jeresig",  
7   dataType: "jsonp",  
8   jsonpCallback: "logResults"  
9 });  
10 ~  
11 ~  
12 ~  
13 ~  
14 ~  
15 ~  
16 ~  
17 ~  
18 ~  
19 ~  
20 ~  
21 ~
```

The code is highlighted with syntax colors: functions in brown, variables in blue, and strings in green. The terminal status bar at the bottom shows "logResults.jsonp - 9 lines" and "162 bytes written to disk". The status bar also indicates "Line 1/9 Col 1".

JSX files formatting highlighting support.

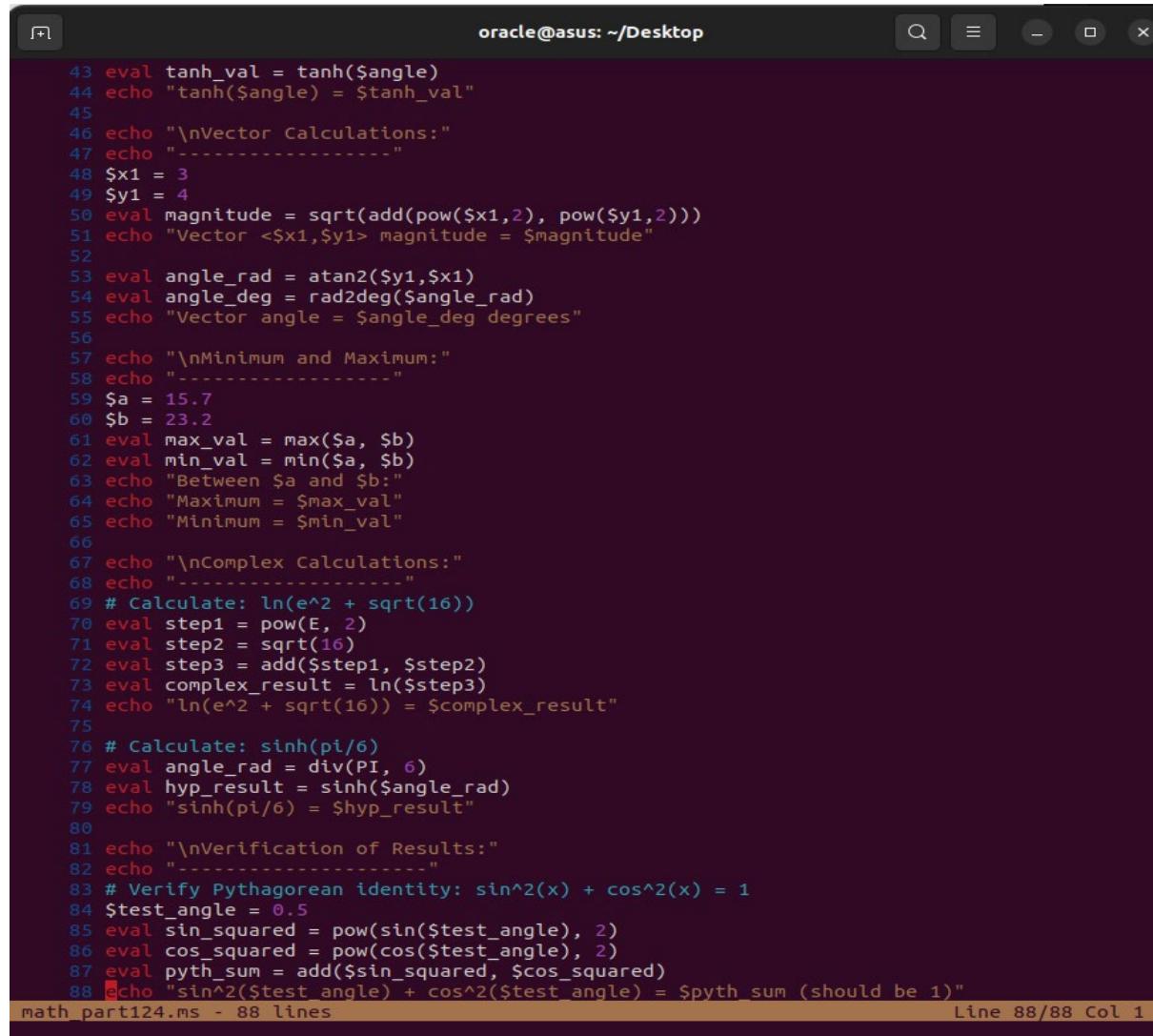


A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window displays a file named "MyComponent.jsx" with the following content:

```
1 import React from 'react';
2
3 const Button = (props) => {
4   return <button onClick={props.onClick}>{props.text}</button>;
5 }
6
7 const MyComponent = () => {
8   return (
9     <div>
10       <Button text="Click me!" onClick={() => alert('Button clicked!')} />
11     </div>
12   );
13 };
14
15 export default MyComponent;
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
```

The code uses color-coded syntax highlighting: red for keywords like `import`, `const`, and `export`, blue for strings and numbers, and green for comments. Line numbers are shown on the left. The terminal window has a dark background and includes status bars at the top and bottom.

Mshell scripts highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains a shell script named "math_part124.ms" with line numbers 43 to 88. The script performs various mathematical calculations using eval and echo commands. Syntax highlighting is applied to different parts of the code, such as variable names (\$angle, \$x1, \$y1, \$magnitude, \$a, \$b, \$max_val, \$min_val), function names (sqrt, atan2, rad2deg, max, min, ln, pow, add, sinh, PI), and operators (+, -, *, /). The background of the terminal is dark, and the highlighted text is in various colors like blue, green, and orange. The status bar at the bottom shows "math_part124.ms - 88 lines" and "Line 88/88 Col 1".

```
43 eval tanh_val = tanh($angle)
44 echo "tanh($angle) = $tanh_val"
45
46 echo "\nVector Calculations:"
47 echo -----
48 $x1 = 3
49 $y1 = 4
50 eval magnitude = sqrt(add(pow($x1,2), pow($y1,2)))
51 echo "Vector <$x1,$y1> magnitude = $magnitude"
52
53 eval angle_rad = atan2($y1,$x1)
54 eval angle_deg = rad2deg($angle_rad)
55 echo "Vector angle = $angle_deg degrees"
56
57 echo "\nMinimum and Maximum:"
58 echo -----
59 $a = 15.7
60 $b = 23.2
61 eval max_val = max($a, $b)
62 eval min_val = min($a, $b)
63 echo "Between $a and $b:"
64 echo "Maximum = $max_val"
65 echo "Minimum = $min_val"
66
67 echo "\nComplex Calculations:"
68 echo -----
69 # Calculate: ln(e^2 + sqrt(16))
70 eval step1 = pow(E, 2)
71 eval step2 = sqrt(16)
72 eval step3 = add($step1, $step2)
73 eval complex_result = ln($step3)
74 echo "ln(e^2 + sqrt(16)) = $complex_result"
75
76 # Calculate: sinh(pi/6)
77 eval angle_rad = div(PI, 6)
78 eval hyp_result = sinh($angle_rad)
79 echo "sinh(pi/6) = $hyp_result"
80
81 echo "\nVerification of Results:"
82 echo -----
83 # Verify Pythagorean identity: sin^2(x) + cos^2(x) = 1
84 $test_angle = 0.5
85 eval sin_squared = pow(sin($test_angle), 2)
86 eval cos_squared = pow(cos($test_angle), 2)
87 eval pyth_sum = add($sin_squared, $cos_squared)
88 echo "sin^2($test_angle) + cos^2($test_angle) = $pyth_sum (should be 1)"
```

Php language highlighting support.

A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following PHP code:

```
1 <?php
2 class Car {
3     function Car() {
4         $this->model = "Tesla";
5     }
6 }
7
8 // create an object
9 $Lightning = new Car();
10
11 // show object properties
12 echo $Lightning->model;
13 ?>
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
```

The code is highlighted with syntax colors: blue for tags like <?php, class, function, echo, and new; red for strings ("Tesla") and variables (\$Lightning); and orange for comments (//). The terminal status bar at the bottom shows "tesla.php - 13 lines" and "Line 8/13 Col 20".

Phtml markup language highlighting support.

```
oracle@asus: ~/Desktop
```

```
1 <?php
2 $unique = uniqid();
3 $reference = function($lang) use ($unique) {
4     return $lang . $unique;
5 };
6 ?>
7 <div class="code-snippets">
8     <ul class="nav nav-tabs">
9         <?php $first = true; foreach($examples as $lang => $code):?>
10            <li role="presentation" class="<?php echo $reference($lang) . ($first ? ' active' : '')?> $first
11            <?php endforeach ?>
12        </ul>
13        <div class="tab-content">
14            <?php $first = true; foreach($examples as $lang => $code): ?>
15            <div role="tabpanel" class="tab-pane <?php echo ($first ? 'active' : '')?> $first = false ?>" id=
16                <pre><code><?php echo $code ?></code></pre>
17            </div>
18            <?php endforeach ?>
19        </div>
20    </div>
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
```

```
examples.phtml - 20 lines
891 bytes written to disk
```

```
Line 13/20 Col 1
```

Py3 language files formatting highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following Python code:

```
1 # Program to display calendar of the given month and year
2
3 # importing calendar module
4 import calendar
5
6 yy = 2025 # year
7 mm = 11 # month
8
9 # To take month and year input from the user
10 # yy = int(input("Enter year: "))
11 # mm = int(input("Enter month: "))
12
13 # display the calendar
14 print(calendar.month(yy, mm))
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
```

The terminal also displays the file information at the bottom:

toa.py3 - 14 lines
319 bytes written to disk

Line 8/14 Col 10

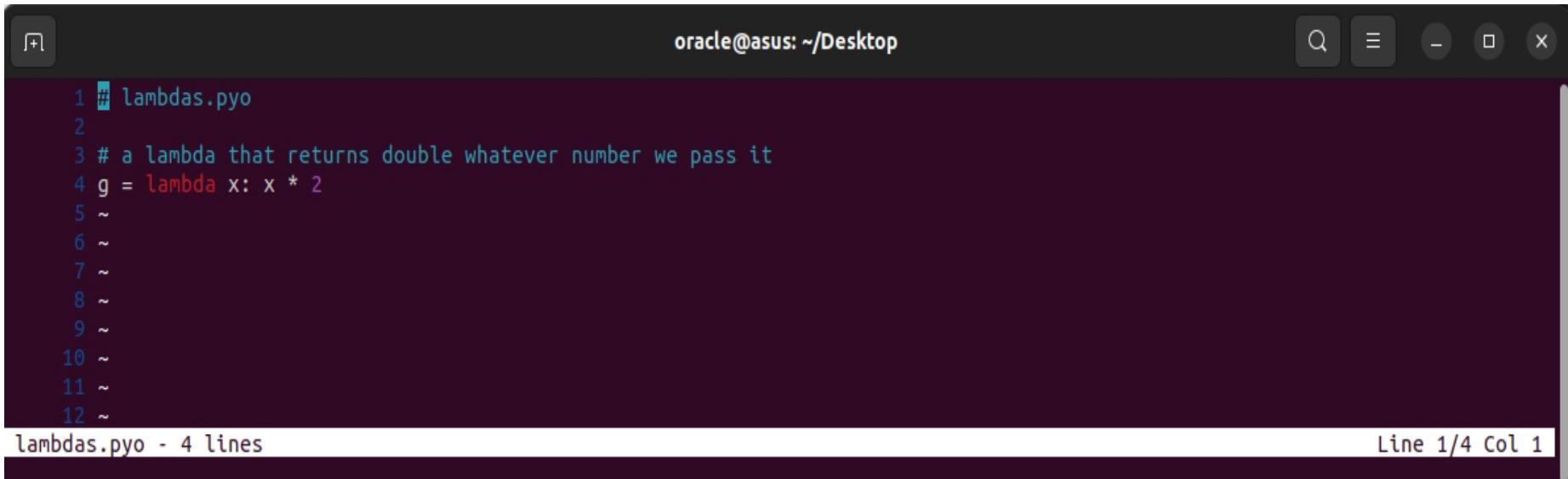
Pyc language files formatting highlighting support.

A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following code:

```
1 # my_module.pyc
2 def my_function():
3     print("Hello, world!")
4
5 # main.py
6 import my_module
7
8 # This will cause Python to recompile my_module.py and generate a new my_module.pyc file
9 # the next time main.py is run
10 ~
11 ~
12 ~
13 ~
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
```

The code is color-coded: blue for numbers, red for keywords like `def` and `print`, and orange for strings. The terminal status bar at the bottom shows "my_module.pyc - 9 lines" and "Line 1/9 Col 1". It also includes a help message: "Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)".

Pyo language files formatting highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following Python code:

```
1 # lambdas.pyo
2
3 # a lambda that returns double whatever number we pass it
4 g = lambda x: x * 2
5 ~
6 ~
7 ~
8 ~
9 ~
10 ~
11 ~
12 ~
```

The code is highlighted with syntax coloring: numbers are blue, strings are light blue, and the lambda keyword is red. The terminal status bar at the bottom shows "lambdas.pyo - 4 lines" on the left and "Line 1/4 Col 1" on the right.

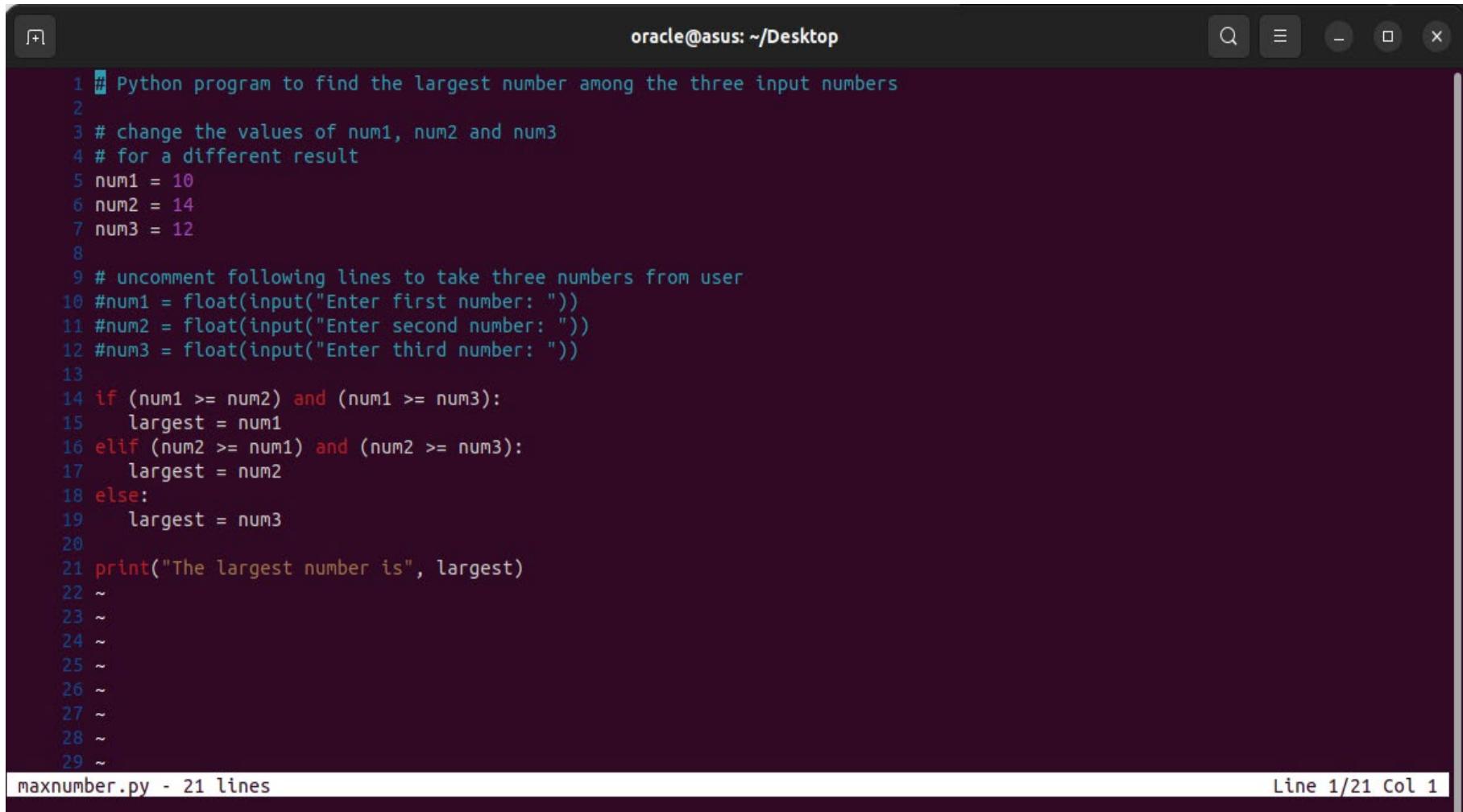
Pyw language files formatting highlighting support.

```
oracle@asus: ~/Desktop
```

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3
4 "Sample gui2py application"
5
6 from __future__ import with_statement    # for python 2.5 compatibility
7
8 __author__ = "Igor Lukyanov (igor.lukyanov@appservgrid.com)"
9 __copyright__ = "Copyright (C) 2013- Igor Lukyanov"
10 __license__ = "LGPL 3.0"
11
12 # images were taken from Pythoncard's proof and widgets demos
13 # for more complete examples, see each control module
14
15 import datetime      # base imports, used by some controls and event handlers
16 import decimal
17 import time
18
19 import gui          # import gui2py package (shortcuts)
20
21 # set default locale to handle correctly numeric format (maskedit):
22 import wx, locale
23 #locale.setlocale(locale.LC_ALL, u'es_ES.UTF-8')
24 if wx.VERSION < (2, 9, 5) or 'classic' in wx.version():
25     loc = wx.Locale(wx.LANGUAGE_DEFAULT, wx.LOCALE_LOAD_DEFAULT)
26
27 # --- here go your event handlers ---
28
29 def load(evt):
30
31     # load the list items with sample data:
32     lv = mywin['listview']
33     lv.items = [[str(i), chr(i)*5] for i in range(65, 92)]
34
35     # load the tree with sample data:
36     tv = mywin['treeview']
37     root = tv.items.add(text="Root")
textui.pyw - 260 lines
```

Line 27/260 Col 9

Python language files formatting highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains Python code for finding the largest number among three inputs. The code uses a combination of comments and user input to determine the largest number. The terminal interface includes standard window controls (minimize, maximize, close) and a status bar at the bottom indicating the file name and line/col information.

```
1 # Python program to find the largest number among the three input numbers
2
3 # change the values of num1, num2 and num3
4 # for a different result
5 num1 = 10
6 num2 = 14
7 num3 = 12
8
9 # uncomment following lines to take three numbers from user
10 #num1 = float(input("Enter first number: "))
11 #num2 = float(input("Enter second number: "))
12 #num3 = float(input("Enter third number: "))
13
14 if (num1 >= num2) and (num1 >= num3):
15     largest = num1
16 elif (num2 >= num1) and (num2 >= num3):
17     largest = num2
18 else:
19     largest = num3
20
21 print("The largest number is", largest)
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
```

maxnumber.py - 21 lines Line 1/21 Col 1

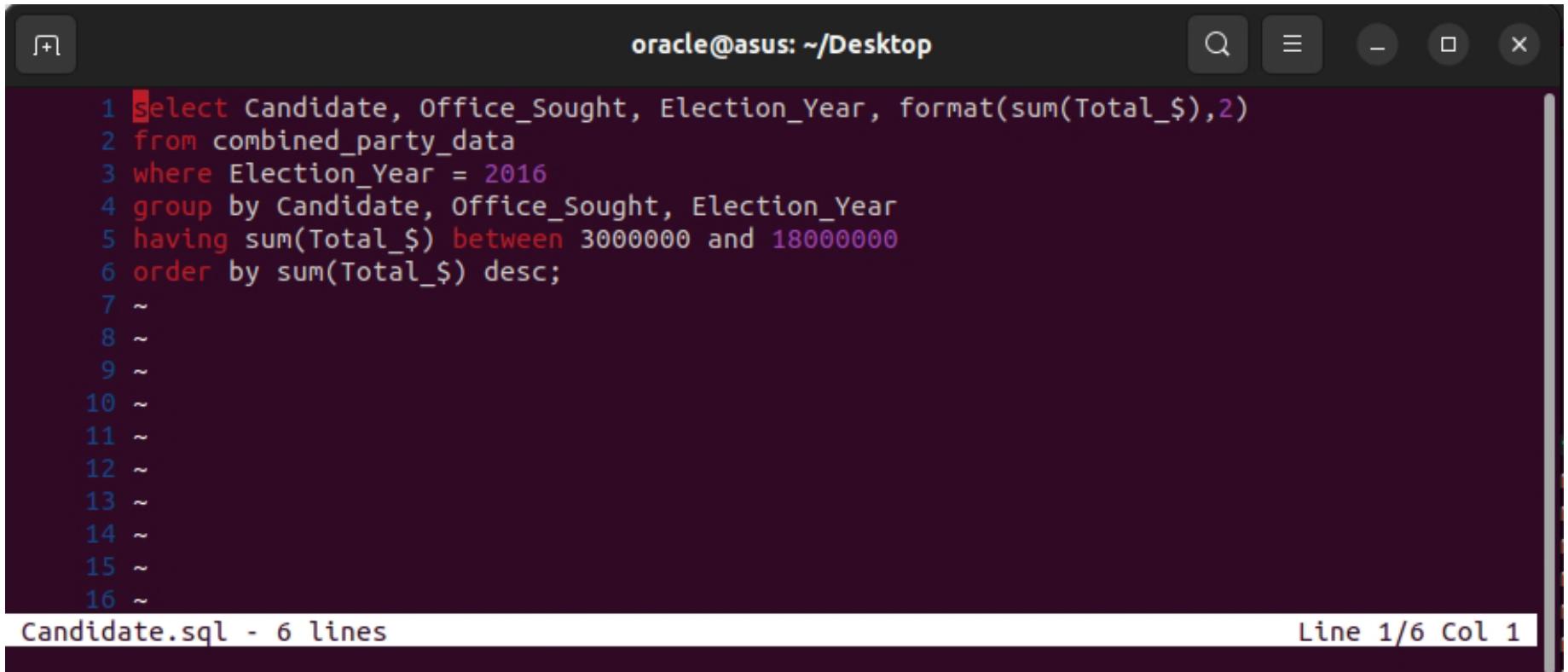
Ruby language files formatting highlighting support.

A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window has a dark theme with light-colored text. The terminal displays the following Ruby code:

```
1 def alternating_characters?(s)
2   type = /[aeiou]/, /[^aeiou]/].cycle
3
4   if s.start_with?(/[^aeiou]/)
5     type.next
6   end
7
8   s.chars.all? { |ch| ch.match?(type.next) }
9 end
10
11 alternating_characters?("ateciyu")
12 # true
13 ~
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
```

The code defines a function `alternating_characters?` that checks if a string consists of alternating vowels and consonants. It uses regular expressions to cycle between matching and non-matching patterns. The terminal also shows the command `alternative_characte - 12 lines` at the bottom left and "Line 7/12 Col 1" at the bottom right.

SQL language files formatting highlighting support.



A screenshot of a terminal window titled "oracle@asus: ~/Desktop". The window contains the following SQL query:

```
1 select Candidate, Office_Sought, Election_Year, format(sum(Total$_),2)
2 from combined_party_data
3 where Election_Year = 2016
4 group by Candidate, Office_Sought, Election_Year
5 having sum(Total$_) between 3000000 and 18000000
6 order by sum(Total$_) desc;
7 ~
8 ~
9 ~
10 ~
11 ~
12 ~
13 ~
14 ~
15 ~
16 ~
```

The code is color-coded: "select", "from", "where", "group", "having", "order" are in red; "Candidate", "Office_Sought", "Election_Year", "format", "sum", "between", "desc" are in blue; numbers and the decimal separator are in purple; and the placeholder "_" is in green. The terminal status bar at the bottom shows "Candidate.sql - 6 lines" on the left and "Line 1/6 Col 1" on the right.

XML files formatting highlighting support.

```
oracle@asus: ~/Desktop
1 <widget>
2     <debug>on</debug>
3     <window title="Sample Konfabulator Widget">
4         <name>main_window</name>
5         <width>500</width>
6         <height>500</height>
7     </window>
8     <image src="Images/Sun.png" name="sun1">
9         <hOffset>250</hOffset>
10        <vOffset>250</vOffset>
11        <alignment>center</alignment>
12    </image>
13    <text data="Click Here" size="36" style="bold">
14        <name>text1</name>
15        <hOffset>250</hOffset>
16        <vOffset>100</vOffset>
17        <alignment>center</alignment>
18        <onMouseUp>
19            sun1.opacity = (sun1.opacity / 100) * 90;
20        </onMouseUp>
21    </text>
22 </widget>
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
widget.xml - 22 lines
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)
Line 1/22 Col 1
```

CLI options of Mel Editor.

OPTIONS

Option

Option	Action
-h --help	Prints the help
-v --version	Prints the version of mel
-b --backup	Create backup (.bak) file before saving
-l --line <number> <file_name>	Open file with cursor on specified line number
-w --width <columns>	Set visual column width marker

CLI options of Mel Editor ('mel -h' or 'mel - -help').

OPTIONS

Option

Action

-h --help	Prints the help
-v --version	Prints the version of mel
-b --backup	Create backup (.bak) file before saving
-l --line <number> <file_name>	Open file with cursor on specified line number
-w --width <columns>	Set visual column width marker

CLI options of Mel Editor ('mel -h' or 'mel - --help').

```
oracle@asus:~$ mel -h
```



```
oracle@asus:~$ mel -h
Usage: mel [OPTIONS] [FILE]

KEYBINDINGS
-----
Keybinding      Action
Ctrl-Q          Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S          Save, requires input of file name, if file didn't exist
Ctrl-F          Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching
Ctrl-N          Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R          Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-J          Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input
Ctrl-G          Go to line Number, requires input the line number
Ctrl-B          Hide/Show line numbering
ctrl-E          Flip line upwards
ctrl-D          Flip line downwards
Ctrl-C          Copy line
ctrl-X          Cut line
Ctrl-V          Paste line
Ctrl-Z          Undo
Ctrl-Y          Redo
Ctrl-P          Pause mel (type "fg" to resume)
Ctrl-W          Retrieve Ollama LLM response
Ctrl-H          Toggle this help screen
Home           Move the cursor to the beginning of the line
End             Move cursor to end of line
PgUp            Up page scroll
PgDn            Down page scroll
Up               Move cursor up one position
Down             Move cursor down one position
Left              Move cursor left one position
Right             Move cursor right one position
Backspace        Delete character

OPTIONS
-----
option          Action
-h | --help      Prints the help
-v | --version   Prints the version of mel
-b | --backup    Create backup (.bak) file before saving
-l | --line <number> <file_name>  Open file with cursor on specified line number
-w | --width <columns>   Set visual column width marker
-----
Supports highlighting for C,C++,Java,Bash,Mshell,Python,PHP,Javascript,JSON,XML,SQL,Ruby,Go
License: Public domain libre software GPL3,v.0.2.0, 2025
Initial coding: Igor Lukyanov, igor.lukyanov@appservgrid.com
For now, usage of UTF-8 is recommended.
oracle@asus:~$
```

CLI options to check version of Mel Editor ('mel -v' or 'mel - --version').

```
oracle@asus:~/Desktop$ mel --version
mel - version 0.2.0
oracle@asus:~/Desktop$
```

CLI option to make backup of editing file ('mel -b name_of_editing_file' or 'mel --backup name_of_editing_file').

- CLI option to make backup of editing file ('mel -b name_of_editing_file' or 'mel --backup name_of_editing_file'). Backup of initial editing file is creating after changes and first saving of editing file.
- A backup of the original file is created with the suffix .bak

```
/home/igor/bren3277 > mel -b bren3277.c
```



```
total 112
drwxrwxr-x  2 igor igor  4096 Feb  1 17:54 .
drwxr-xr-x 50 igor igor  4096 Feb  1 17:50 ..
-rw-rw-r--  1 igor igor 50490 Feb  1 17:54 bren3277.c
-rw-rw-r--  1 igor igor 50489 Feb  1 17:54 bren3277.c.bak
```

CLI option to make backup of editing file ('mel -b name_of_editing_file' or 'mel --backup name_of_editing_file').

- CLI option to make backup of editing file ('mel -b name_of_editing_file' or 'mel --backup name_of_editing_file'). Backup of initial editing file is creating after changes and first saving of editing file.
- A backup of the original file is created with the suffix .bak

```
/home/igor/bren3277 > mel -b bren3277.c
```



```
total 112
drwxrwxr-x  2 igor igor  4096 Feb  1 17:54 .
drwxr-xr-x 50 igor igor  4096 Feb  1 17:50 ..
-rw-rw-r--  1 igor igor 50490 Feb  1 17:54 bren3277.c
-rw-rw-r--  1 igor igor 50489 Feb  1 17:54 bren3277.c.bak
```

CLI option to open file on specified line number ('mel -l line_number name_of_editing_file' or 'mel --line line_number name_of_editing_file').

```
/home/igor/bren3277 > mel -l 1273 bren3277.c
```



The terminal window shows the command:

```
oracle@asus: ~/Desktop
```

The code editor displays the C file `bren3277.c` with line 1273 highlighted. The status bar at the bottom indicates:

```
bren3277.c - 1582 lines
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)
Line 1273/1582 Col 1
```

CLI option to open file on specified line number ('mel -w width_marker_number name_of_editing_file' or 'mel --width width_marker_number name_of_editing_file').

```
/home/igor/bren3277 > mel -w 82 bren3277.c
```



```
oracle@asus: ~/Desktop
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5 #include <readline/readline.h>
6 #include <readline/history.h>
7 #include <cctype.h>
8 #include "parser.h"
9 #include "lexer.h"
10 #include "variables.h"
11 #include <<JSON.h>>
12 #include <stdbool.h>
13 #include <fcntl.h>
14 #include <sys/wait.h>
15 #include <math.h>
16 #include <glob.h>
17 #include <time.h>
18
19 #define MAX_LINE_LEN 1024
20 #define MAX_BLOCK_LEN (MAX_LINE_LEN * 10)
21 #define HISTORY_FILE ".mshellhistory"
22 #define MAX_MODELS 4
23 #define MAX_COMMAND_LEN 1024
24 #define MAX_CONFIG_LEN 512
25 #define MAX_CACHE_SIZE 200
26 #define MAX_FILE_SIZE 4096
27 #define PROMPT_SIZE 2048
28
29 typedef struct {
30     char role[16];
31     char content[4096];    // Increased from 1024 to handle command outputs
32     char command[1024];   // Added for exec modes
33 } Message;
34
35 // Global cache and configuration variables
36 Message message_cache_ollama1[MAX_CACHE_SIZE];
37 Message message_cache_ollama2[MAX_CACHE_SIZE];
bren3277.c - 1582 lines
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)
```

Line 1/1582 Col 1

Multiple CLI options - I.

```
/home/igor/bren3277 > mel -b -l 124 -w 80 bren3277.c
```

```
oracle@asus:~$ mel -b -l 87 aaa.py
```

```
oracle@asus:~$ mel -b -w 80 aaa.py
```

```
oracle@asus:~$ mel -l 45 -w 83 aaa.py
```

Multiple CLI options - II.

```
oracle@asus:~$ mel -b -l 45 -w 83 aaa.py
```



```
oracle@asus: ~
1 import os
2 from pdf2image import convert_from_path
3 import music21
4 from PIL import Image
5 import cv2
6 import numpy as np
7
8 def pdf_to_images(pdf_path, output_folder):
9     """
10     Convert PDF pages to images.
11     """
12     images = convert_from_path(pdf_path)
13     image_paths = []
14     for i, image in enumerate(images):
15         output_path = os.path.join(output_folder, f"page_{i + 1}.png")
16         image.save(output_path, "PNG")
17         image_paths.append(output_path)
18     return image_paths
19
20 def recognize_music(image_path):
21     """
22     Recognize musical notation in the image and create a Music21 score.
23     (Placeholder function - replace with an actual OMR library like Audiveris)
24     """
25     # Example placeholder logic: creates a basic C major scale
26     score = music21.stream.Stream()
27     for note_name in ['C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4', 'C5']:
28         score.append(music21.note.Note(note_name, quarterLength=1))
29     return score
30
31 def generate_midi(score, output_midi_path):
32     """
33     Generate a MIDI file from a Music21 score.
34     """
35     score.write('midi', fp=output_midi_path)
36     print(f'MIDI file created: {output_midi_path}')
37
38 def main():
39     pdf_path = 'input.pdf' # Replace with your PDF file path
40     output_folder = 'output_images'
41     os.makedirs(output_folder, exist_ok=True)
42
43     # Convert PDF to images
44     image_paths = pdf_to_images(pdf_path, output_folder)
45     print(f'PDF converted to images: {image_paths}')
46
47     # Process each image and generate MIDI
48     for i, image_path in enumerate(image_paths):
49         print(f'Processing image: {image_path}')
50         score = recognize_music(image_path)
```

How to load content for editing (file) – I.

```
/home/igor/mshell/src > mel mshell.c
```



```
oracle@asus: ~
1 import os
2 from pdf2image import convert_from_path
3 import music21
4 from PIL import Image
5 import cv2
6 import numpy as np
7
8 def pdf_to_images(pdf_path, output_folder):
9     """
10     Convert PDF pages to images.
11     """
12     images = convert_from_path(pdf_path)
13     image_paths = []
14     for i, image in enumerate(images):
15         output_path = os.path.join(output_folder, f"page_{i + 1}.png")
16         image.save(output_path, "PNG")
17         image_paths.append(output_path)
18     return image_paths
19
20 def recognize_music(image_path):
21     """
22     Recognize musical notation in the image and create a Music21 score.
23     (Placeholder function - replace with an actual OMR library like Audiveris)
24     """
25     # Example placeholder logic: creates a basic C major scale
26     score = music21.stream.Stream()
27     for note_name in ['C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4', 'C5']:
28         score.append(music21.note.Note(note_name, quarterLength=1))
29     return score
30
31 def generate_midi(score, output_midi_path):
32     """
33     Generate a MIDI file from a Music21 score.
34     """
35     score.write('midi', fp=output_midi_path)
36     print(f'MIDI file created: {output_midi_path}')
37
38 def main():
39     pdf_path = 'input.pdf' # Replace with your PDF file path
40     output_folder = 'output_images'
41     os.makedirs(output_folder, exist_ok=True)
42
43     # Convert PDF to images
44     image_paths = pdf_to_images(pdf_path, output_folder)
45     print(f'PDF converted to images: {image_paths}')
46
47     # Process each image and generate MIDI
48     for i, image_path in enumerate(image_paths):
49         print(f'Processing image: {image_path}')
50         score = recognize_music(image_path)
```

How to load content for editing (pipe) – II.

```
/home/igor > cat .bashrc | mel
```



```
1 # ~/.bashrc: executed by bash(1) for non-login shells.
2 # See /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
3 # for examples
4
5 # If not running interactively, don't do anything
6 case $- in
7     *i*) ;;
8     *) return;;
9 esac
10
11 # don't put duplicate lines or lines starting with space in the history.
12 # See bash(1) for more options
13 HISTCONTROL=ignoreboth
14
15 # append to the history file, don't overwrite it
16 shopt -s histappend
17
18 # for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
19 HISTSIZE=1000
20 HISTFILESIZE=2000
21
22 # check the window size after each command and, if necessary,
23 # update the values of LINES and COLUMNS.
24 shopt -s checkwinsize
25
26 # If set, the pattern "##" used in a pathname expansion context will
27 # match all files and zero or more directories and subdirectories.
28 #shopt -s globstar
29
30 # make less more friendly for non-text input files, see lesspipe(1)
31 [ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"
32
33 # set variable identifying the chroot you work in (used in the prompt below)
34 if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
35     debian_chroot=$(cat /etc/debian_chroot)
36 fi
37
```

[No Name] - 117 lines
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)

Line 1/117 Col 1

How to load content for editing (commands) – III.

```
/home/igor > (df -m; ls -la; inxi) | less
```



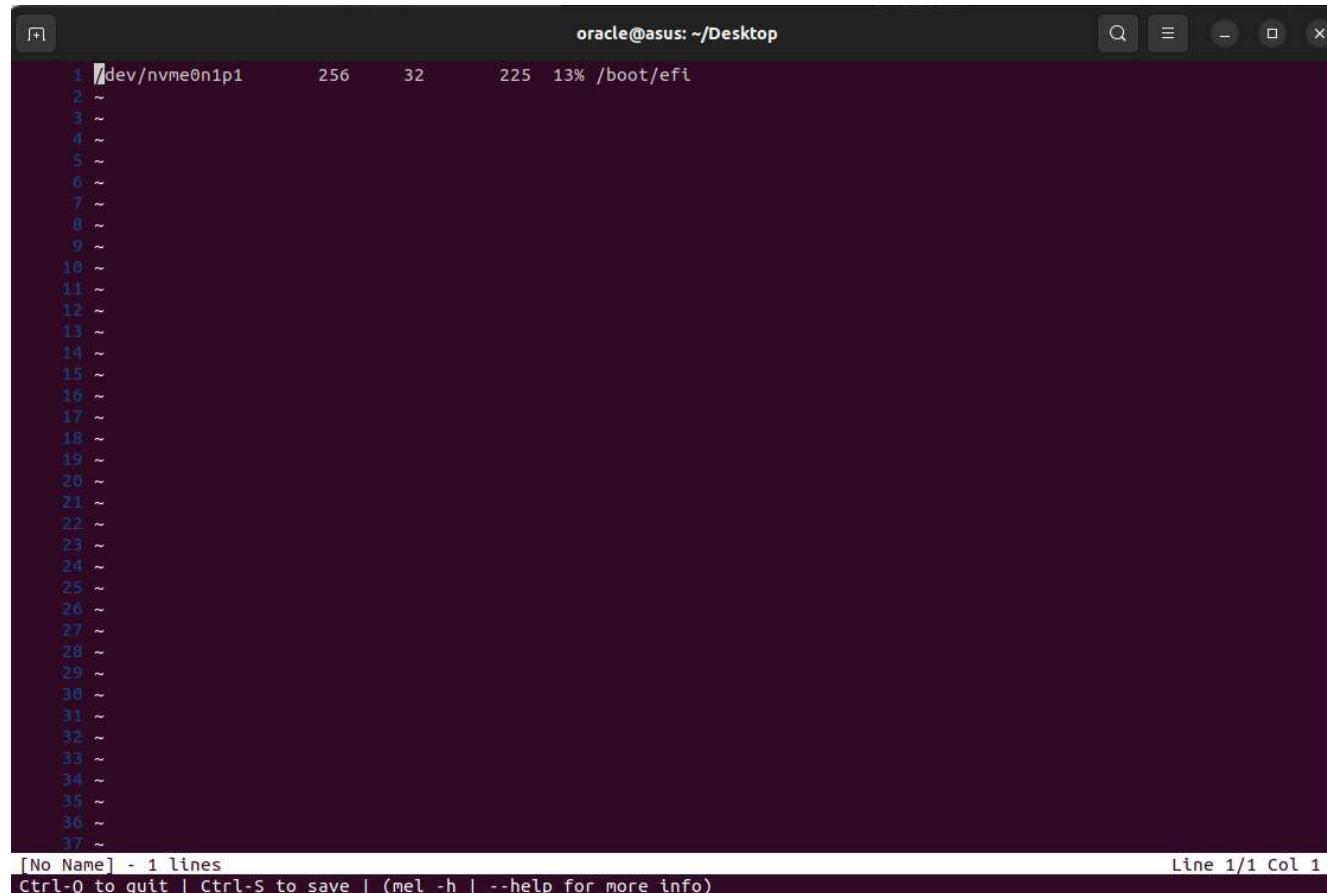
```
48 drwxr-xr-x 4 igor igor 4096 Jan 25 23:09 mshell_backup_January25_2025_fixation
49 drwxr-xr-x 4 igor igor 4096 Jan 9 11:57 mshell_backup_January9_2025
50 drwxr-xr-x 4 igor igor 4096 Jan 25 16:55 mshell_backup_January9_2025_lua_fixed
51 drwxrwxr-x 4 igor igor 4096 Feb 1 00:09 mshell_Feb1_work_on_statements
52 -rw-r--r-- 1 igor igor 57004 Feb 3 11:59 .mshellhistory
53 drwxr-xr-x 4 igor igor 4096 Jan 26 16:37 mshell_jan26_v020
54 drwxr-xr-x 4 igor igor 4096 Jan 26 16:38 mshell_jan26_v020_with_debug
55 drwxr-xr-x 4 igor igor 4096 Jan 27 04:03 mshell_jan27_fixation
56 drwxr-xr-x 4 igor igor 4096 Jan 27 13:19 mshell_jan27_prompts_path_time_fixation
57 drwxr-xr-x 4 igor igor 4096 Jan 27 13:50 mshell_jan27_while_statement_fixation
58 drwxr-xr-x 4 igor igor 4096 Jan 27 05:17 mshell_jan27_wildcards_fixation
59 drwxr-xr-x 4 igor igor 4096 Jan 22 07:50 mshell_January22_beforecall_ollama_change
60 drwxr-xr-x 4 igor igor 4096 Jan 22 22:57 mshell_January23_propbelswithollamapipes
61 drwxr-xr-x 4 igor igor 4096 Jan 23 04:12 mshell_January23_propbelswithollamapipes_2
62 drwxr-xr-x 4 igor igor 4096 Jan 23 14:34 mshell_January23_propbelswithollamapipes_3
63 drwxr-xr-x 4 igor igor 4096 Jan 24 17:48 mshell_January24_lua_fixation
64 drwxr-xr-x 4 igor igor 4096 Jan 9 21:22 mshell_January9_9pm
65 drwxr-xr-x 4 igor igor 4096 Jan 14 16:06 mshell_ollamaexec
66 drwxr-xr-x 4 igor igor 4096 Jan 14 20:12 mshell_ollamaexec2
67 drwxr-xr-x 4 igor igor 4096 Jan 17 23:32 mshell_pipesfixation_forinternalcommands_january17_2025
68 drwxr-xr-x 4 igor igor 4096 Jan 18 00:24 mshell_pipesfixation_forinternalcommands_withdebug_january17_2025
69 -rw-r--r-- 1 igor igor 816 Jan 28 22:28 .mshellrc
70 -rw-r--r-- 1 igor igor 725 Jan 26 02:22 .mshellrc_old
71 drwxr-xr-x 4 igor igor 4096 Jan 7 13:30 mshell_real_backup
72 drwxr-xr-x 3 igor igor 4096 Jan 6 15:53 mshell_without_lua_support
73 -rwxr--r-- 1 root root 196865 Jan 30 05:34 mshell.zip
74 drwxr-xr-x 2 igor igor 4096 Jan 28 16:27 .ollama
75 -rw-rw-r-- 1 igor igor 1579 Jan 17 17:27 planets.json
76 -rw-r--r-- 1 igor igor 807 Dec 30 05:06 .profile
77 -rw----- 1 igor igor 7 Jan 23 19:40 .python_history
78 drwx----- 3 igor igor 4096 Jan 27 01:09 snap
79 drwx----- 2 igor igor 4096 Feb 1 01:21 .ssh
80 -rw-r--r-- 1 igor igor 0 Jan 7 17:40 .sudo_as_admin_successful
81 drwxr-xr-x 2 root root 4096 Jan 30 04:14 test_highlighting
82 CPU: quad core Intel Core i5-8265U (-MT MCP-) speed/min/max: 798/400/3900 MHz
83 Kernel: 6.8.0-52-generic x86_64 Up: 1d 14h 24m Mem: 3294.0/19714.9 MiB (16.7%)
84 Storage: 6.83 TiB (2.7% used) Procs: 337 Shell: sh inxi: 3.3.13
```

[No Name] - 84 lines

Line 77/84 Col 1

How to load content for editing (commands) – IIIa.

```
/home/igor > (df -m | grep /boot/efi) | mel
```



```
oracle@asus: ~/Desktop
1 /dev/nvme0n1p1      256     32      225  13% /boot/efi
2 ~
3 ~
4 ~
5 ~
6 ~
7 ~
8 ~
9 ~
10 ~
11 ~
12 ~
13 ~
14 ~
15 ~
16 ~
17 ~
18 ~
19 ~
20 ~
21 ~
22 ~
23 ~
24 ~
25 ~
26 ~
27 ~
28 ~
29 ~
30 ~
31 ~
32 ~
33 ~
34 ~
35 ~
36 ~
37 ~
[No Name] - 1 lines
Ctrl-Q to quit | Ctrl-S to save | (mel -h | --help for more info)
Line 1/1 Col 1
```

How to load content from llm's with supporting context – IIIb.

```
/home/igor/mel > ollama2 "What makes function time at C language?" | mel
```



```
oracle@asus: ~/Desktop
1 DEBUG_EXEC: Got command: ollama2 "What makes function time at C language?"
2 The term "function time" in C programming typically refers to the execution time of a specific function
3 or block of code within a program. This can be measured and analyzed to understand performance characteristics,
4 optimize code, or identify bottlenecks.
5
6 In C, you can measure function execution time using various methods. One common approach is to use high-resolution timers
7 provided by the operating system or libraries such as `<time.h>` on Unix-like systems. Here's a simple example of how
8 to measure the execution time of a function:
9
10 ````c
11 #include <stdio.h>
12 #include <time.h>
13
14 // Function whose execution time we want to measure
15 void myFunction() {
16     // Simulate some computations
17     for (int i = 0; i < 1000000; i++) {
18         int j = i * i;
19     }
20 }
21
22 int main() {
23     clock_t start, end;
24     double cpu_time_used;
25
26     start = clock(); // Record the start time
27
28     myFunction(); // Call the function whose execution time we want to measure
29
30     end = clock(); // Record the end time
31
32     cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC; // Calculate the elas
33 DEBUG_PIPE: Child 1: Command completed
34 ~
35 ~
36 ~
37 ~
123.txt - 33 lines
1198 bytes written to disk
```

Line 8/33 Col 1

Installation from binaries.

KEYBINDINGS

Keybinding	Action
Ctrl-Q	Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S	Save, requires input of file name, if file didn't exist
Ctrl-F	Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching
Ctrl-N	Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R	Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-J	Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input
Ctrl-G	Go to line Number, requires input the line number
Ctrl-B	Hide/Show line numbering
Ctrl-E	Flip line upwards
Ctrl-D	Flip line downwards
Ctrl-C	Copy line
Ctrl-X	Cut line
Ctrl-V	Paste line
Ctrl-Z	Undo
Ctrl-Y	Redo
Ctrl-P	Pause mel (type "fg" to resume)
Ctrl-W	Retrieve Ollama LLM response
Ctrl-H	Toggle this help screen
Home	Move the cursor to the beginning of the line
End	Move cursor to end of line
PgUp	Up page scroll
PgDn	Down page scroll
Up	Move cursor up one position
Down	Move cursor down one position
Left	Move cursor left one position
Right	Move cursor right one position
Backspace	Delete character

Keybindings.

KEYBINDINGS

Keybinding	Action
Ctrl-Q	Exit, 3 times click Ctrl-Q if file was changed without saving
Ctrl-S	Save, requires input of file name, if file didn't exist
Ctrl-F	Search by pattern, Esc - exit from Search, Enter and Arrows to interact searching
Ctrl-N	Forward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-R	Backward Search by pattern after Ctrl-F. Esc - exit from Search, works after Ctrl-F only
Ctrl-J	Global replacement of character combinations, Input Search and Replace patterns, Esc to cancel, Enter to input
Ctrl-G	Go to line Number, requires input the line number
Ctrl-B	Hide/Show line numbering
Ctrl-E	Flip line upwards
Ctrl-D	Flip line downwards
Ctrl-C	Copy line
Ctrl-X	Cut line
Ctrl-V	Paste line
Ctrl-Z	Undo
Ctrl-Y	Redo
Ctrl-P	Pause mel (type "fg" to resume)
Ctrl-W	Retrieve Ollama LLM response
Ctrl-H	Toggle this help screen
Home	Move the cursor to the beginning of the line
End	Move cursor to end of line
PgUp	Up page scroll
PgDn	Down page scroll
Up	Move cursor up one position
Down	Move cursor down one position
Left	Move cursor left one position
Right	Move cursor right one position
Backspace	Delete character

License.

```
1 /*  
2 *  
3 *  
4 * This program is free software: you can redistribute it and/or modify  
5 * it under the terms of the GNU General Public License as published by  
6 * the Free Software Foundation, either version 3 of the License, or  
7 * any later version.  
8 *  
9 * This program is distributed in the hope that it will be useful,  
0 * but WITHOUT ANY WARRANTY; without even the implied warranty of  
1 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
2 * GNU General Public License for more details.  
3 *  
4 * You should have received a copy of the GNU General Public License  
5 * along with this program. If not, see <http://www.gnu.org/licenses/>.  
6 */
```

Questions & Answers

Contact information:

Igor Lukyanov, Art2Dec SoftLab

- E-mail support for MEL editor: igor101964@gmail.com
- NIC-handle: LIK26
- WWW: <http://www.appservgrid.com>
- E-resume: <http://www.art2dec.co/paw1/>

Art2Dec®