

# 프로그래밍언어

# 에디터와 컴파일

담당교수: 임현승

TA: 지수환(93suhwan@gmail.com)

# Editor

---

- Editor란?
  - 코드 작성을 편리하게 도와주는 프로그램
  - e.g.) Vim, GNU Emacs, etc.
  - eclipse, MS Visual Studio 등 편집기를 내장한 통합개발환경(IDE)도 있음
- Editor를 사용하는 이유
  - 빠른 문서 편집
  - 편리한 단축키
  - 마우스 없이 키보드만을 사용하여 문서 편집 가능

```
#include <stdio.h>

int main(void)
{
    printf("Hello, World!");
}
```

```
#include <stdio.h>

int main(void)
{
    printf("Hello, World!");
}
```

# IDE

(Integrated Development Environment)

# OCaml-Top

해당 라인 실행



The screenshot shows the OCaml-Top REPL interface. The left pane displays the source code:

```
→ 3 + 4;;  
→ let message = "Hello world!";;  
→ print_endline message;;
```

The right pane shows the execution results:

```
OCaml version 4.01.0+ocp1  
# 3 + 4  
- : int = 7  
# let message = "Hello world!"  
val message : string = "Hello world!"  
# print_endline message  
Hello world!  
- : unit = ()
```

소스 코드

결과

# OCaml-Top Command

- 단축키(출처: <http://www typerex.org/ocaml-top.html>)

## File management

|               |                               |
|---------------|-------------------------------|
| <b>Ctrl n</b> | Start with a new (blank) file |
| <b>Ctrl o</b> | Load a file                   |
| <b>Ctrl s</b> | Save the current file         |

## Edition

|                     |       |
|---------------------|-------|
| <b>Ctrl c</b>       | Copy  |
| <b>Ctrl x</b>       | Cut   |
| <b>Ctrl v</b>       | Paste |
| <b>Ctrl z</b>       | Undo  |
| <b>Ctrl Shift z</b> | Redo  |

## Execution

|               |                                |
|---------------|--------------------------------|
| <b>Ctrl e</b> | Execute the current expression |
| <b>Escape</b> | Stop ongoing execution         |

## General

|               |                   |
|---------------|-------------------|
| <b>Ctrl +</b> | Zoom in           |
| <b>Ctrl -</b> | Zoom out          |
| <b>F4</b>     | Select font       |
| <b>F5</b>     | Switch colors     |
| <b>F11</b>    | Switch fullscreen |
| <b>Ctrl q</b> | Quit              |

# vim(vi) 편집기 ([참고1](#), [참고2](#))

---

- vi란?

- Emacs와 함께 유닉스 환경에서 가장 많이 쓰이는 문서 편집기
- 한 화면을 편집하는 비주얼 에디터(visual editor)라는 뜻에서 유래
- 간결하지만, 강력한 기능으로 열광적인 사용자 보유

- 조작법

- 프로그램 시작 or Esc키를 누를 때 명령(normal) 모드,
- I키를 누르면 편집(insert) 모드로 동작
- Esc키를 누를 때(명령)까지 문서 작성을 할 수 있음
- 편집모드에서만 내용을 넣거나 수정할 수 있음

# Emacs 편집기 [\(참고\)](#)

---

- Emacs
  - 사용자가 많은 부분을 설정할 수 있는 고성능 문서 편집기
  - 현재 유닉스 환경을 주된 대상으로 하고 있으나, MS Windows 등 다양한 환경에서도 이용 가능
- 조작법
  - vi와 다르게 키보드의 배치와 **상관 없이** 조작 명령을 갖는 것이 특징
    - e.g.) 상하좌우 이동은 각각 ctrl+p, n, b, f(Previous, Next, Back, and Forward)
  - vi에 있는 편집 모드, 명령 모드 등이 **없음**
  - vi-mode, viper-mode와 같이 Emacs에서도 vi의 조작을 모방할 수 있음

# Editor Command

---

( C = Control, M = Alt )

|         | Vim | Emacs   |
|---------|-----|---------|
| 한 문자 삭제 | x   | C-d     |
| 한 줄 삭제  | dd  | C-a C-k |
| 한 줄 복사  | yy  | M-w     |
| 한 단어 삭제 | dw  | M-d     |
| 한 단어 복사 | yw  |         |
| 붙여넣기    | p   | C-y     |



# Editor Command

( C- = Control, M- = Alt )

|           | Vim         | Emacs           |
|-----------|-------------|-----------------|
| 드래그       | v           | Shift 누른 채로 방향키 |
| 커서위치에서 입력 | i(명령 -> 편집) |                 |
| 다음 줄에서 입력 | o(명령 -> 편집) |                 |
| 편집 -> 명령  | Esc         |                 |
| 검색 하기     | /[word]     | C-s             |
| 저장        | :w [file]   | C-x C-s         |
| 종료        | :q          | C-x C-c         |

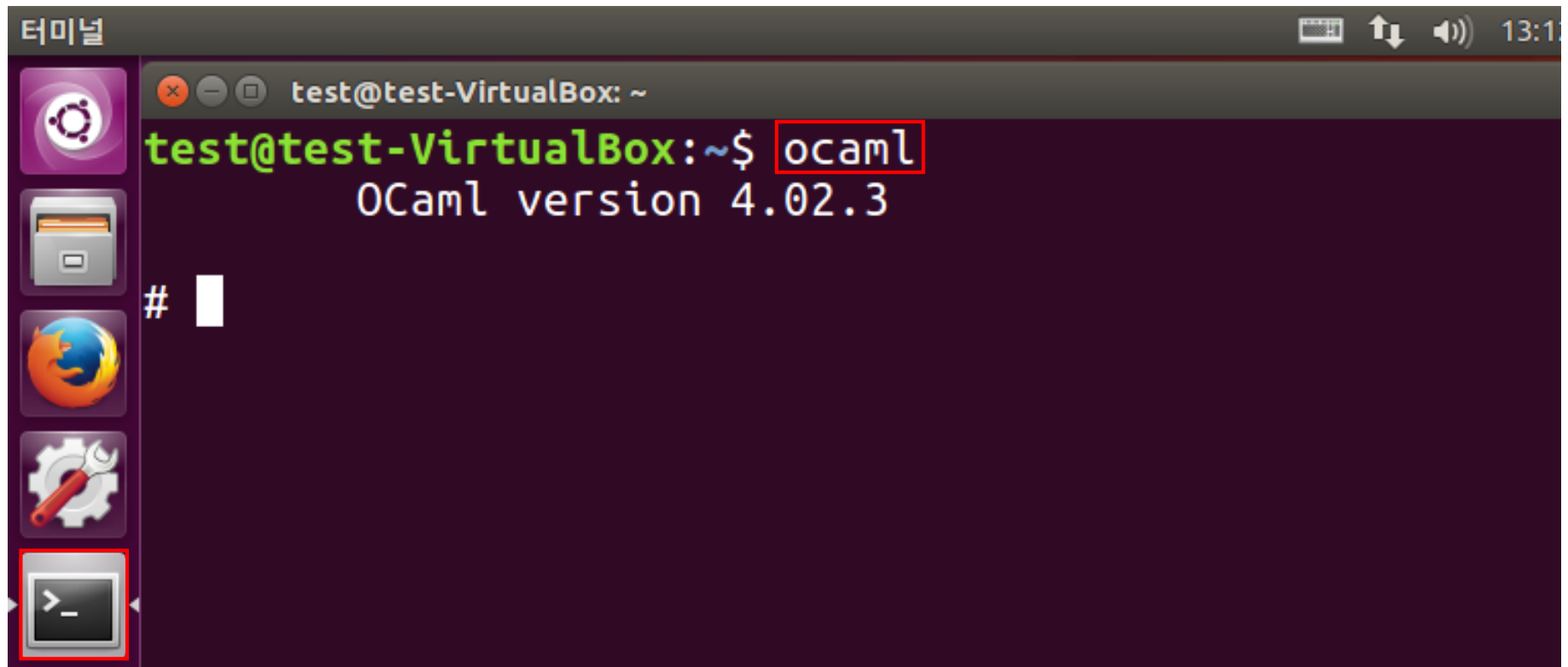
# Terminal Command (Ubuntu/Linux)

---

|                       |                                  |
|-----------------------|----------------------------------|
| 디렉토리 내 파일(디렉토리) 목록 보기 | ls                               |
| 디렉토리 이동               | cd [dir_name]      (../ 이전 디렉토리) |
| 파일 실행                 | ./file_name      (./ 현재 디렉토리)    |
| 디렉토리 만들기              | mkdir dir_name                   |
| 파일, 디렉토리 삭제           | rm [-r]      (-r = 디렉토리 삭제)      |

해석기(Interpreter)

# 해석기

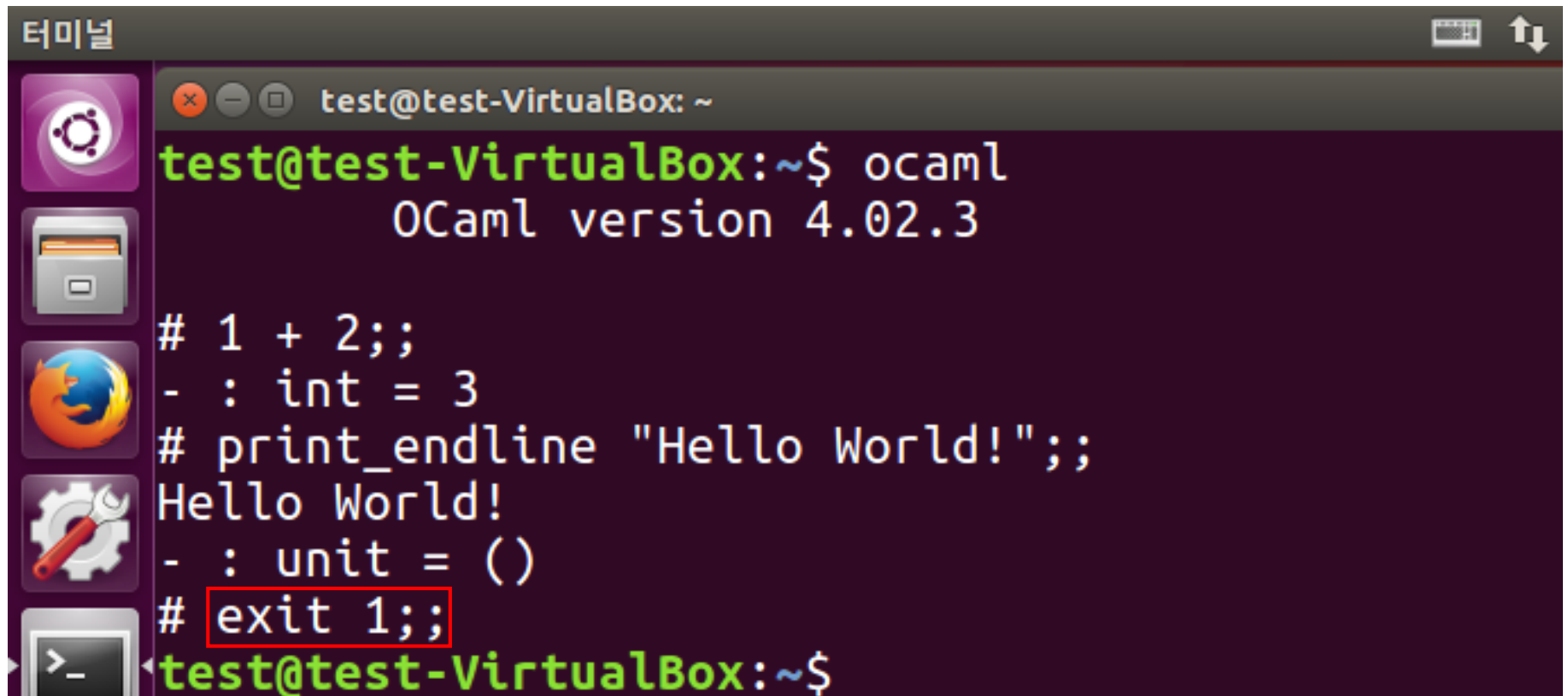


The screenshot shows a terminal window titled "터미널" (Terminal) with a dark background. The window's title bar includes standard window controls and system icons like a keyboard, mouse, and volume. On the left side, there is a vertical dock with icons for a terminal, file manager, Firefox, settings, and a terminal icon which is highlighted with a red rectangle. The terminal content shows the prompt "test@test-VirtualBox: ~" followed by the command "ocaml" (highlighted with a red rectangle) and its output "OCaml version 4.02.3". A new prompt line "# " is visible below the output.

```
test@test-VirtualBox: ~$ ocaml
OCaml version 4.02.3
#
```

좌측의 상태표시줄 5번째 터미널 실행  
ocaml 입력

# 해석기

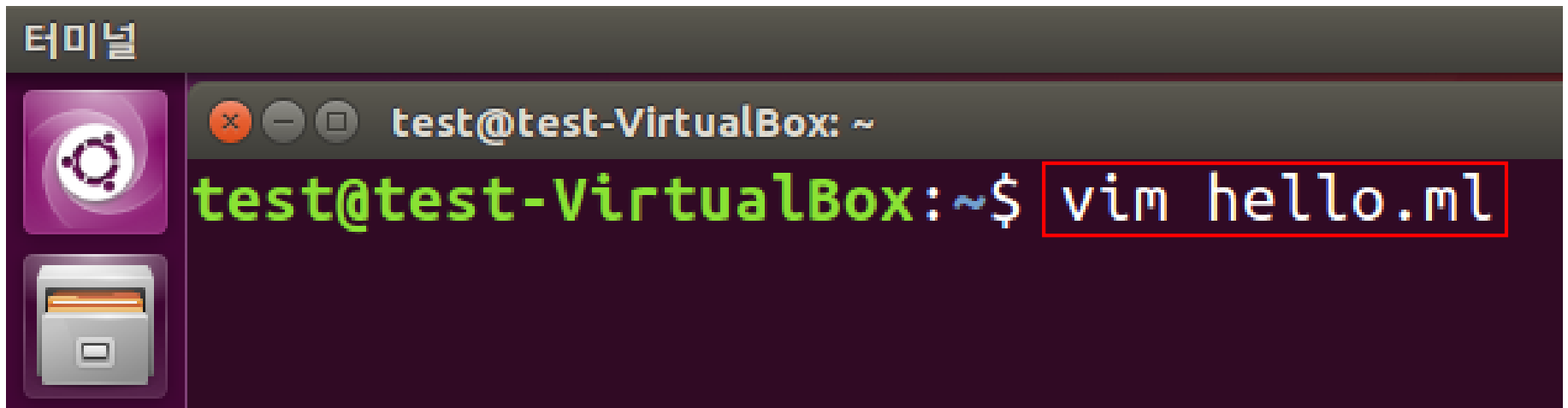
A terminal window titled '터미널' (Terminal) with a dark background. The window shows the execution of OCaml code. The prompt is 'test@test-VirtualBox: ~'. The code entered is: 'ocaml', 'OCaml version 4.02.3', '# 1 + 2;;', '- : int = 3', '# print\_endline "Hello World!";;', 'Hello World!', '- : unit = ()', and '# exit 1;'. The 'exit 1;' line is highlighted with a red box. The prompt returns to 'test@test-VirtualBox: ~\$'.

```
터미널
test@test-VirtualBox: ~
test@test-VirtualBox:~$ ocaml
OCaml version 4.02.3
# 1 + 2;;
- : int = 3
# print_endline "Hello World!";;
Hello World!
- : unit = ()
# exit 1;;
test@test-VirtualBox:~$
```

종료 방법 : `exit [int];;`

컴파일(Compile)

# 컴파일

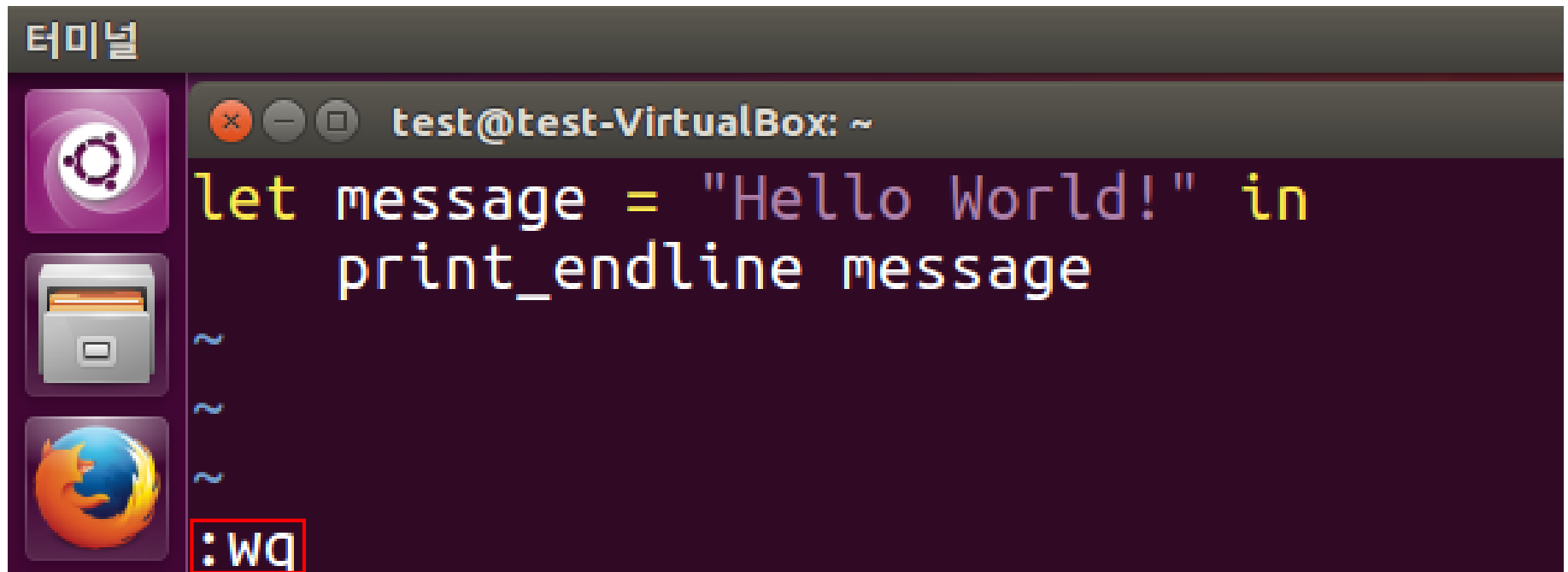


vi(m) [파일 이름]

emacs [파일 이름]

# 컴파일

터미널



```
test@test-VirtualBox: ~  
let message = "Hello World!" in  
  print_endline message  
~  
~  
~  
:wq
```

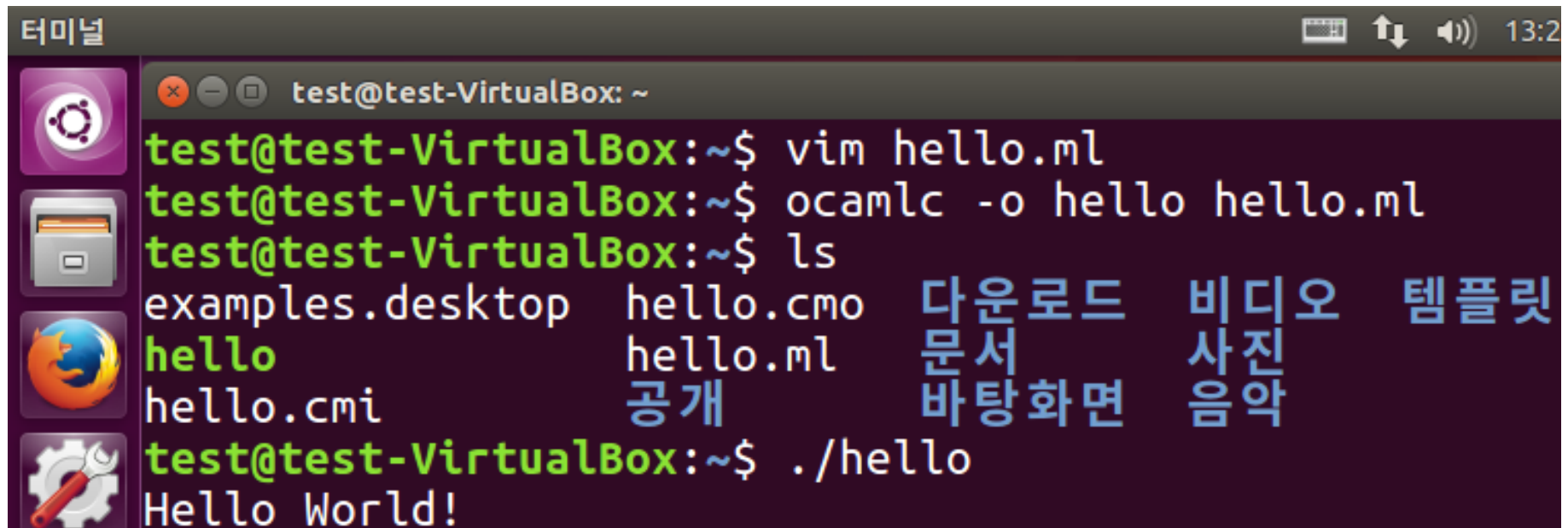
":" 콜론을 반드시 입력해야 함

저장 :w , 종료 :q

저장 & 종료 :wq



# 컴파일



A terminal window titled '터미널' (Terminal) with a dark background. The window shows a series of commands and their outputs in a green monospace font. The commands are: `vim hello.ml`, `ocamlc -o hello hello.ml`, `ls`, and `./hello`. The output of `ls` lists files: `examples.desktop`, `hello.cmo`, `hello`, `hello.cmi`, and `hello.ml`. The output of `./hello` is `Hello World!`. On the right side of the terminal, there are four vertical columns of Korean text: '다운로드' (Download), '비디오' (Video), '템플릿' (Template), '문서' (Document), '사진' (Photo), '공개' (Public), '바탕화면' (Desktop), and '음악' (Music). The terminal window has standard Linux window controls (close, maximize, minimize) and a system tray at the top right showing icons for a keyboard, a mouse, and a speaker, along with the time '13:2'.

```
test@test-VirtualBox: ~  
test@test-VirtualBox:~$ vim hello.ml  
test@test-VirtualBox:~$ ocamlc -o hello hello.ml  
test@test-VirtualBox:~$ ls  
examples.desktop  hello.cmo  hello      hello.cmi  hello.ml  
test@test-VirtualBox:~$ ./hello  
Hello World!
```

컴파일 방법 : `ocamlc [-o 실행파일 이름] [소스파일 이름]`  
[-o 실행파일 이름]을 생략 할 경우 `a.out` 이 실행파일  
실행 방법 : `./[실행파일 이름]`

# 출력함수

---

- 화면에 변수를 출력해주는 함수
  - Top\_level과 다르게 출력 함수가 필요함
  - print\_string s;;
    - 문자열 타입의 s를 화면에 출력
  - print\_int n;;
    - 정수형 타입의 n을 화면에 출력
  - print\_float f;;
    - 부동소수 타입의 f를 화면에 출력
  - print\_endline s;;
    - 문자열 타입의 s를 화면에 출력하고 다음 줄로 커서 이동
  - print\_newline ();;
    - 화면의 커서의 위치를 다음 줄로 이동

# 출력함수

- 소스 코드

```
터미널
test@test-VirtualBox: ~
print_newline ();;
print_string "STRING";;
print_int 1;;
print_float 1.;;
print_endline "ENDLINE!";;
```

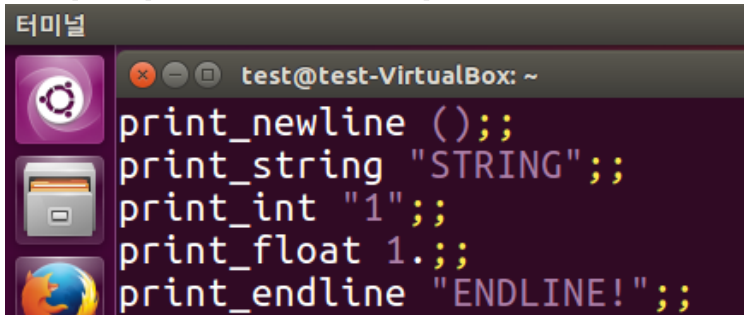
- 실행 결과

```
터미널
test@test-VirtualBox: ~
test@test-VirtualBox:~$ vim print.ml
test@test-VirtualBox:~$ ocamlc print.ml
test@test-VirtualBox:~$ ls
a.out          print.cmo      다운로드  비디오  템플릿
examples.desktop print.ml        문서
print.cmi      공개           바탕화면  음악
test@test-VirtualBox:~$ ./a.out

STRING11.ENDLINE!
```

# 컴파일 에러 체크

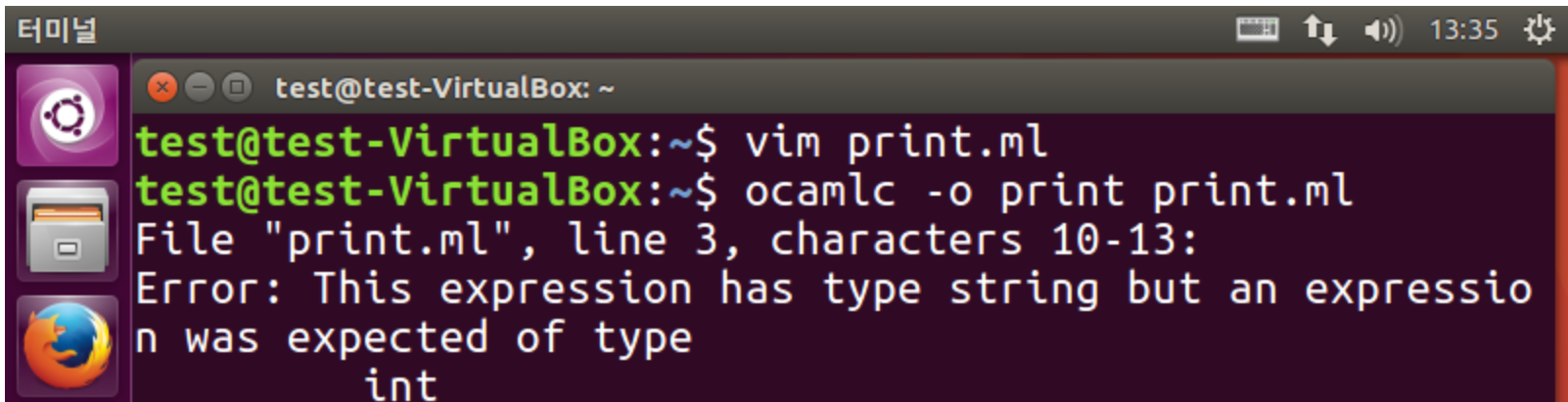
- 에러 코드 예



A terminal window titled 'test@test-VirtualBox: ~' showing the following OCaml code:

```
print_newline ();;  
print_string "STRING";;  
print_int "1";;  
print_float 1.;;  
print_endline "ENDLINE!";;
```

- 컴파일 실행



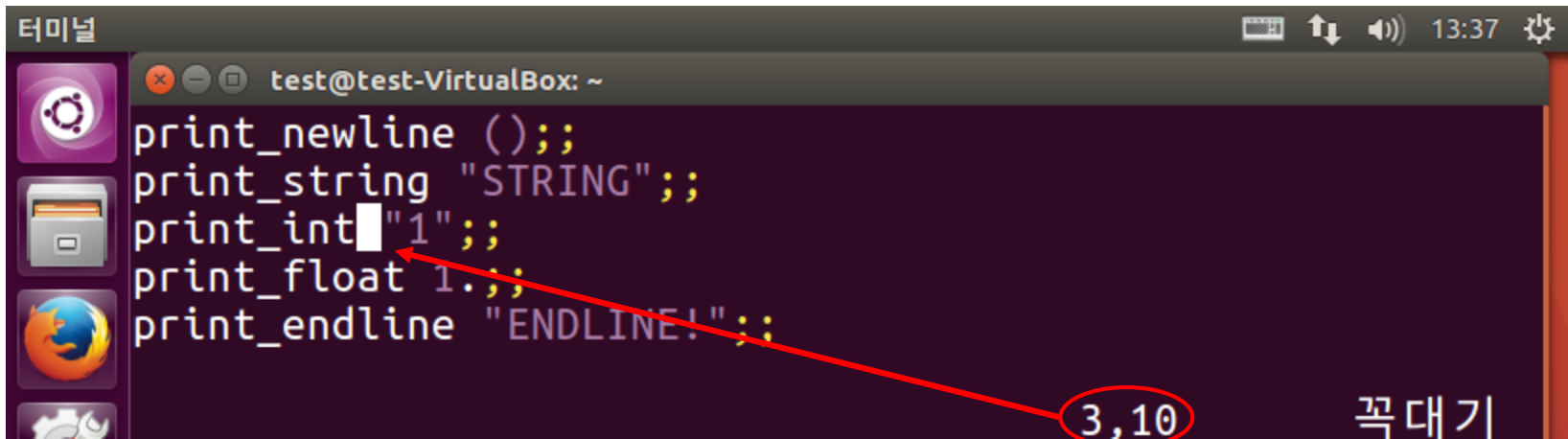
A terminal window titled 'test@test-VirtualBox: ~' showing the compilation process and an error message:

```
test@test-VirtualBox:~$ vim print.ml  
test@test-VirtualBox:~$ ocamlc -o print print.ml  
File "print.ml", line 3, characters 10-13:  
Error: This expression has type string but an expression  
      was expected of type  
              int
```

- 메시지를 통해 3번째 줄의 10-13 구간에서 에러를 확인 할 수 있음
  - int 타입 위치에 string타입을 사용해서 나타난 에러

# 컴파일 에러 체크

- vim에서 에러 위치 확인

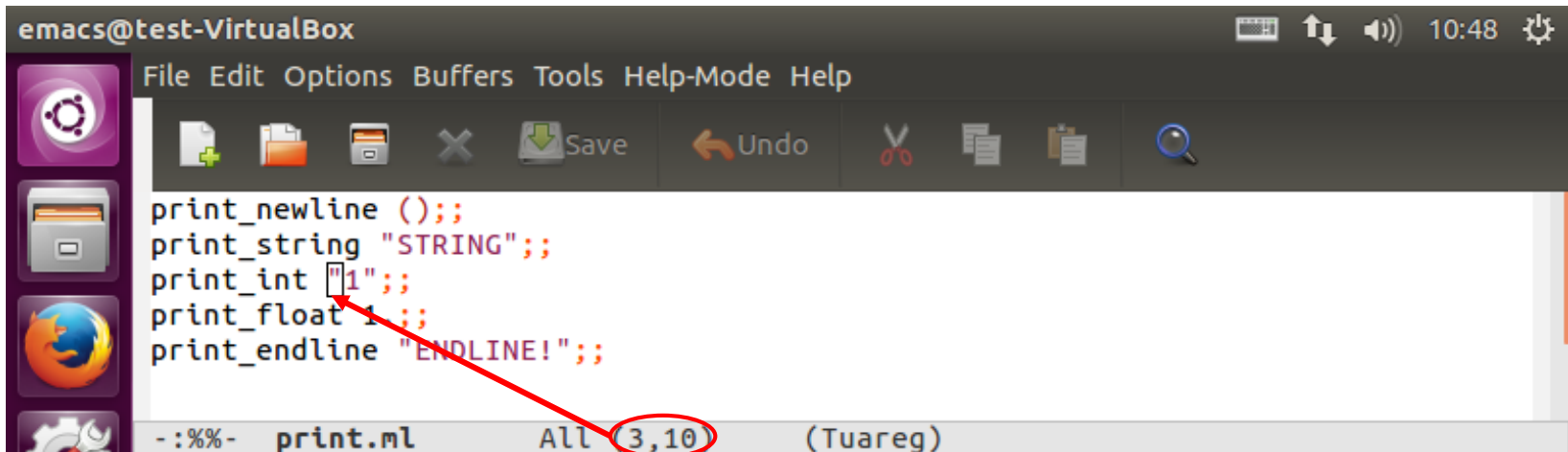


터미널

```
test@test-VirtualBox: ~  
print_newline ();;  
print_string "STRING";;  
print_int "1";;  
print_float 1.;;  
print_endline "ENDLINE!";;
```

3,10 꼭대기

- emacs에서 에러 위치 확인



emacs@test-VirtualBox

File Edit Options Buffers Tools Help-Mode Help

```
print_newline ();;  
print_string "STRING";;  
print_int "1";;  
print_float 1.;;  
print_endline "ENDLINE!";;
```

print.ml All (3,10) (Tuareg)