El editor vi

- Editor muy poderoso y grande
- Solo se describen las características principales
- Algunos comandos pueden parecer problematicos
- Proporciona muchas formas de realizar tareas de edición básicas
- Sugerencia: aprender comandos edición básicos y e ir aprendiendo otros comandos después

Historia de vi

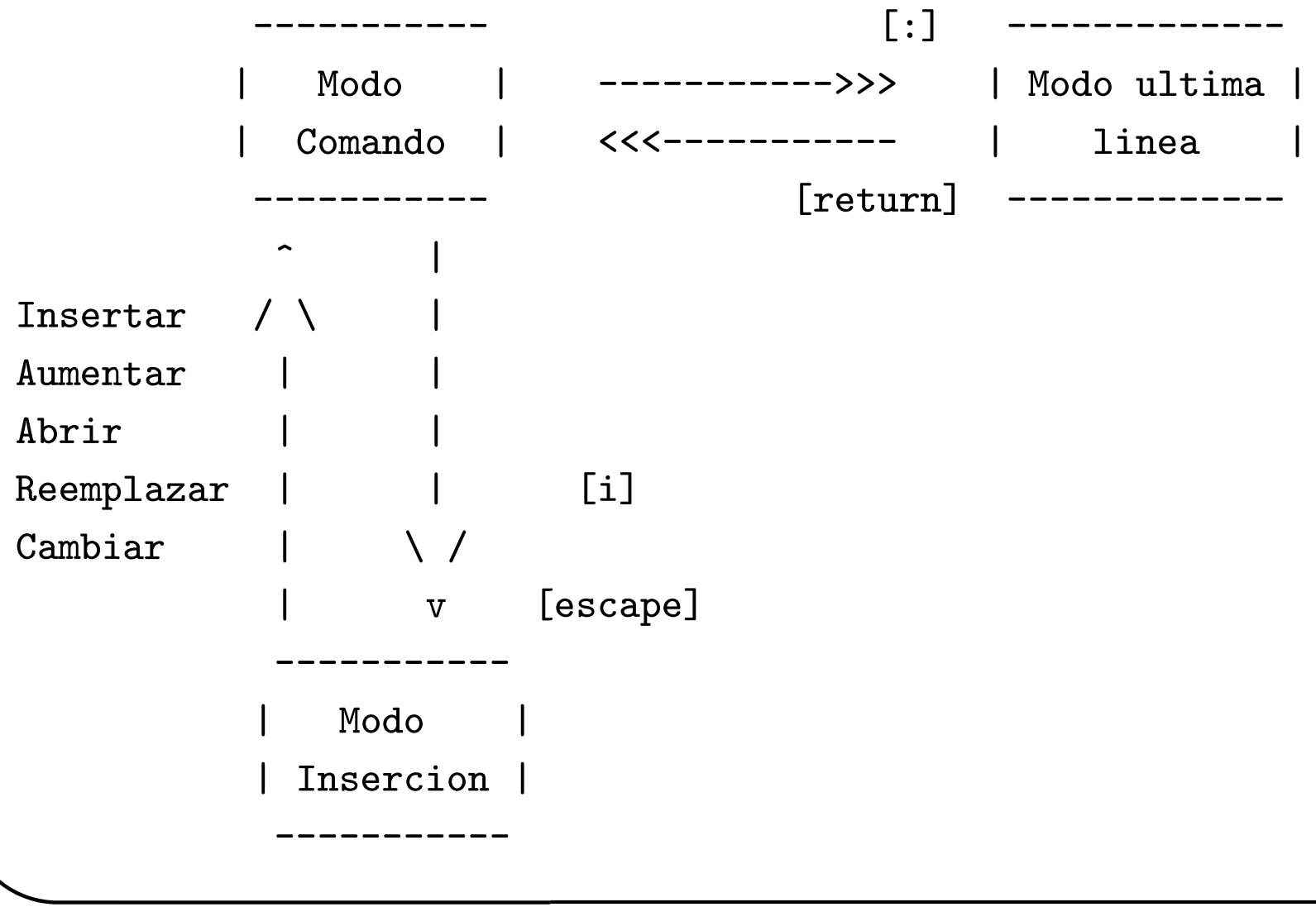
- Desarrollado en la univerdiad de California, Berkely
- Forma parte del Unix de Berkeley BSD
- Antes `vi` el editor estandar de Unix era `ed`
- Editor `ed` era editor de línea, por lo que era complicado ver el contexto de lo que se editaba
- Después apareció editor `ex` un superconjunto de `ed`
- Ventaja `ex`: contaba con una facilidad de despliege que permitía trabajar con una pantalla entera de texto
- La facilidad era llamada a través del comando `vi`

- Usuarios utilizaban dicha facilidad tan seguido que se hizo posible iniciar `ex` de tal forma que se cargara la facilidad desde el inicio
- A la nueva facilidad se le llamó `vi`
- Aún es posible pasar de `ex` a `vi` y viceversa,
  - en `vi` hay que teclear comando `Q` para usar `ex`
  - en `ex` hay que teclear comando `vi`

Modos operación

- Modo comando de `ex`
- Modo inserción de `ex`
- Modo comando de `vi`
- Modo inserción de `vi`
- Modo última línea de `vi`

Relación modos operación



El Buffer de trabajo

- `vi` realiza todo su trabajo en el *buffer de trabajo*
- principio sesión edición traslada archivo del disco al área de trabajo
- durante sesión edición `vi` trabaja sobre la copia
- no cambia contenido hasta que se da el comando de escritura
- cuando se edita un nuevo archivo `vi` no crea el archivo hasra que escribe el contenido del buffer de trabajo en el disco, usualmente al final de la sesión de edición actual
- *desventaja:* si accidentalmente se termina la sesión sin escribir en el disco, todo el trabajo se pierde

- *ventaja:* si se realizaron cambios no deseados es posible abortar la sesión sin escribir en discos
- posible leer un archivo sin modificación:

```
$ view toto
$ vi -R toto
```

Comandos inicio

Comando	Función
`vi archivo`	Editar `archivo` empezando en línea 1
`vi +n archivo`	Editar `archivo` empezando en línea n
`vi + archivo`	Editar `archivo` empezando en última línea
`vi -r archivo`	Editar `archivo` después caída sistema

Unidades de medida

- **caracter:** es un simple caracter, visible o no, incluyendo <espacio> y tabuladores.
- **palabra:** similar a un palabra oridnaria en español
- **línea:** cadena de caracteres rodeada por un <ret>; no es necesariamente una simple línea física de la terminal
- **enunciado:** empieza en el final del enunciado anterior y termina con un parrafo
- **parrafo:** es precedido y segudio por una o más líneas en blanco

Ejemplos unidades de medida

- `perla` ejemplo 1 palabras
- `perla!` ejemplo 2 palabras
- `perla!)` ejemplo 4 palabras
- `perla!)"The` ejemplo 4 palabras
- Ejemplo de 11 palabras:
  `Esta es una linea consisa.`
- Ejemplo de una línea con dos enunciados
  `That's it. This is one sentence.`
- Ejemplo de una línea con tres enunciados
  `Que? Tres enunciados? Una linea?`

- Ejemplo de tres líneas y un enunciado:

```
Este enunciado ocupa
un total
de tres lineas.
```

- Ejemplo de tres parrafos y un enunciado:

```
Aunque esto es visto

como solo un
enunciado

vi lo considera como
tres parrafos.
```

Movimiento del cursor

Comando	Mueve el cursor
`<espacio>`	Espacio a la derecha
`l`	Espacio a la derecha
→	Espacio a la derecha
`h`	Espacio a la izquierda
←	Espacio a la izquierda
`w`	Palabra a la derecha
`W`	Espacio en blanco delimita palabra derecha
`b`	Palabra a la izquierda
`B`	Espacio en blanco delimita palabra izquierda
`$`	Fin de línea

Comando	**Mueve el cursor**
`e`	Fin palabra a la derecha
`E`	Fin espacio en blanco delimita palabra derecha
`0`	Principio de línea
`^`	Principio de línea
`<ret>`	Principio de la siguiente línea
`j`	Abajo una línea
↓	Abajo una línea
`-`	Principio de la línea de arriba (anterior)
`k`	Arriba una línea

Comando	**Mueve el cursor**
↑	Arriba una línea
)	Fin enunciado
(	Principio enunciado
}	Fin parrafo
{	Principio parrafo

Añadiendo texto

Comando	Inserta texto
`i`	Antes cursor
`I`	Antes del primer caracter no blanco de la línea
`a`	Después cursor
`A`	Al final de la línea
`o`	Abre una línea abajo de la línea actual
`O`	Abre una línea arriba de la línea actual
`r`	Reemplaza el carácter actual
`R`	Reemplaza caracteres hasta que se presiona `<esc>`

Borrando texto

Comando	Efecto
`nx`	Borra `n` caracteres empezando por el actual
`nX`	Borra `n` caracteres antes del actual
`dM`	Borra texto especificado por M
`ndd`	Borra el número de líneas especificadas por `n`
`D`	Borra hasta el final de línea

Ejemplos opciones borrado

Comando Borrado	**Acción**
`dw`	Borra hasta el final de la palabra
`dW`	Borra hasta final del blanco que delimita la palabra
`db`	Borra hasta el principio de la palabra
`dB`	Borra hasta principio del blanco delimita la palabra
`dL`	Borra dese el cursor actual hasta el fin de línea
`d(`	Borra hasta el principio del enunciado
`d)`	Borra hasta el final del enunciado
`d{`	Borra hasta el principio del parrafo

Comando Borrado	**Función**
`d}`	Borra hasta el final del parrafo
`d4)`	Borra hasta el final del cuarto enunciado
`d7}`	Borra hasta el séptimo parrafo de los que preceden
	el inicio del parrafo
`dd`	Borra la línea actual
`5dd`	Borra 5 líneas empezando con la actual
`dL`	Borra hasta la última línea en la pantalla
`dH`	Borra hasta la primera línea en la pantalla
`dG`	Borra hasta el final del buffer de trabajo
`d1G`	Borra hasta el principio del buffer de trabajo

Cambiando textos

Comandos dejan `vi` en MODO INSERCIÓN. Se debe presionar `<esc>` para regresar el MODO COMANDO

Comando	**Efecto**
`ns`	Substituye el número de caracteres empezando por `n`
`cM`	Cambia el texto especificado por `M`
`ncc`	Cambia el número de líneas especificado por `n`
`C`	Cambia al final de la línea

Buscando caracteres

En la siguiente lista `rexp` es una expresión regular que puede ser una simple cadena de caracteres.

Comando	**Efecto**
`/rexp<return>`	Busqueda hacia adelante de `rexp`
`?rexp<return>`	Busqueda hacia atras de `rexp`
`n`	Repite exactamente la busqueda original
`N`	Repite la busqueda original en sentido contrario
`/<return>`	Repite busqueda original hacia adelante
`?<return>`	Repite busqueda original hacia atras

Ejemplos busqueda caracteres

Comando	**Resultado**
`/and`	Busca la siguiente ocurrencia del string `and` *Ejemplos:* `sand, and, standard` slander.
`/\<and\>`	Busca la siguiente ocurrencia de la palabra `and` *Ejemplo:* `and`

`\^The`	Busca la siguiente línea que empieza con `The`
	Ejemplos:
	`The...`
	`There...`

Comando	**Resultado**

`/^[0-9][0-9])`	Busca siguiente línea empieza con números de dos dígitos seguidos por un parentesis que cierra
	Ejemplos:
	`77)...`
	`01)...`
	`15)...`

`/\<[adr]`	Busca la siguiente palabra que empieza con un caracter `a`,`d` o `r` *Ejemplos:* `apple, drive, road, argument`

Cambiando caracteres

Se realiza a partir del comando:

`:[direccion]s/cadena-buscada/cadena-nueva[/g]`

Elemento del Comando	**Contiene**
g	Indica un reemplazo global (más de un reemplazo por línea)
`cadena-nueva`	Cadena de caracteres con la se va a reemplazar
`cadena-buscada`	Expresión regular que puede ser una simple cadena de caracteres

Elemento del Comando	**Contiene**
`direccion`	Un número o dos números de línea separados por una coma `.` representa la línea actual `$` representa la última línea `%` en algunas versiones de `vi` representa el archivo entero Es posible usar una marca en lugar de un número de línea

Ejemplos direcciones

Dirección	Porción direccionada
`5`	línea 5
`77,100`	líneas 77 a 100 (inclusive)
`1,.`	principio buffer trabajo hasta línea actual
`.,$`	línea actual hasta final buffer trabajo
`1,$`	todo el buffer trabajo
`%`	todo el buffer trabajo (solo en unas versiones)
`,,.+10`	línea actual hasta las diez siguientes

Ejemplo de cambio de caracteres

Comando	**Resultado**
`s/bigger/biggest`	Reemplaza cadena `bigger` en la línea actual con `biggest` *Ejemplo:* `bigger` → `biggest`

`1,.s/Ch 1/Ch 2/g`	Reemplaza todo ocurrencia cadena `Ch 1`, en líneas anteriores o en actual, con `Ch 2`
	Ejemplos:
	`Ch 1` → `Ch 2`
	`Ch 12` → `Ch 22`

Comando	**Resultado**

`:1,$s/ten/10/g`	Reemplaza todo ocurrencia cadena `ten` por la cadena `10`
	Ejemplos:
	`ten` → `10`
	`often` → `of10`
	`tenant` → `10ant`

`:1,$s/\<ten\>/10/g`	Reemplaza todo ocurrencia de la palabra `ten` por la cadena `10` *Ejemplos:* `ten` → `10` `often` → `often` (no hay cambio)

Comando	**Resultado**

`:.,.+10s/every/each/g`	Reemplaza todo ocurrencia cadena `every` por la cadena `each`, en la línea actual y en las diez siguientes líneas
	Ejemplos:
	`every` → `each`
	`everything` → `eachting`

Ejemplo substitución expresiones regulares

```
int fx(arg1, arg2, arg3)
  int arg1, arg2, arg3;
{
   /* codigo propio a la funcion */
}
main()
{
  int n,m;

  fx(10, &n, 15); printf(%d\n",n);
  fx(30, &n, 13); printf(%d\n",n);
  fx(3,  &n, 13); printf(%d\n",n);
  fx(2,  &n, 1);  printf(%d\n",n);
```

```
fx(4,  &n, 12); printf(%d\n",n);
```

El programa tiene un error, por lo que:

- Se tiene que invertir el primer y tercer argumento de cada llamada de la función `fx()`
- Imposible de una forma directa con Turbo Pascal
- Solución vi: desplazarse hasta el main y teclear:

```
:.,$s/fx(\([^,]*\),\([^,]*\),\([^)]*\))/fx(\3,\2,\1)/g
```

: comienza un comando

. especifica la línea actual

$ especifica el fin del texto

`s/sdfgfd/sdfgsfdg/g` substicuión primer termino por el segundo en todo el documento (debido a la `g`)

```
:.,$s/fx(\([^,]*\),\([^,]*\),\([^)]*\))/fx(\3,\2,\1)/g
```

`\(...\)` memoriza toda la cadena de caracteres

`[^,]*` significa todos los caracteres diferentes de `,`

`[^)]*` significa todos los caracteres diferentes de `)`

`\3` significa el tercer miembro memorizado por la estructura `\(...\)`

Inicalización variable ambiente TERM

- vi muestra una pantalla completa
- vi muestra tantas líneas de texto como sea capaz de visualizar el usuario en la terminal
- tamaño por default: 23 líneas de 80 caracteres c/u, pero puede llegar a 65 líneas de 155 c/u (*noseprint* de SGI)
- debido a diferentes características de las terminales es importante especificar el tipo de terminal configurando la variable de ambinte del shell: `TERM`
- uno de los valores más comunes es el de `vt100` (un modelo de la DEC)

- en bshell:

```
$TERM=vt100
$export TERM
```

- en cshell:

```
% setenv TERM=vt100
```

- en ambos casos es posible incluirlos en los archivos de configuración (`.profile` y `.cshrc`).

Inserción salida de un comando en el texto

- útil poder insertar la salida de un comando del shell en el texto editado
- vi proporciona la posibilidad de ejecutar una orden dentro de vi y substituir la línea actual con la salida de un comando (p.e. la fecha)
- para insertar la fecha dentro de un documento, a partir del modo comando se necesita teclear:
  `:r !date`
- en el lugar donde se encontraba el cursos aparecerá:
  `Thu Jan 27 18:57:29 CST 2000`

Las opciones de vi

- Permiten "personalizar" vi
- Varias opciones estan definidas por default
- Tres tipos de opciones
  1. booleano (on/off o V/F)
  2. argumento numérico
  3. argumento tipo cadena

Opciones más comunes de vi

Opción	Tipo	Default	Significado
autoindet, ai	on/off	noai	(No) inicia cada línea en la misma columna que la línea anterior
autowrite, aw	on/off	noaw	(No) escribe automáticamente los cambios en el buffer antes de ejecutar ciertas órdenes vi
flash	on/off	flash	Emite en pantalla un parpadeo y no un beep

ignorecase, ic	on/off	noic	Mayusculas y minusculas (no) son equivalentes en busquedas
magic	on/off	magic	nomagic ignora los significados especiales de las expresiones regulares excepto `^ . $`
number,nu	on/off	nonu	(No) numera cada línea
shell,sh	cadena	shell	shell ejecuta comandos
showmode, smd	on/off	nosmd	(No) imprime `<<INPUT MODE>>` en psrte inferior derecha de la pantalla en modo de inserción
terse	on/off	noterse	terse proporciona mensajes cortos de error

wrapmargin, wm	numerico	O (off)	Rompe automáticamente las líneas antes del margen de la derecha (indica donde termina la línea)

Fijando opciones

1. Cambiar o fijar opción durante una sesión de edición de vi, para lo cual hay que:

   (a) teclear `:` (dos puntos) durante el modo comando

   (b) ejecutar el comando `set`

   Ejemplo: `:set wrapmargin=15`

2. Crear un archivo `.exrc` donde se almacenen varios comandos `set`. Por ejemplo:

```
$cat .exrc
set wrapmargin=15
$
```

3. Uso de la variable `EXINIT` en archivos de configuración `.profile` o `.login` (según sea el caso)

   - Ejemplo en bourne-shell:

```
EXINIT="set noautoindent ignorecase magic
        wrapmargin=15 number"
```

   - Ejemplo en C-shell:

```
set EXINIT="set noautoindent ignorecase magic
        wrapmargin=15 number"
```

Otros comandos de las opciones

- Para desplegar el valor de una opción

  `:set <nombre-opcion>?`

- Para ver los valores de todas las opciones

  `:set all`

- Para ver los valores de todas las opciones que se han modificado

  `:set`

Los macros de vi

- Permiten combinar una secuencia de comandos de edición en un solo comando
- Dos caracteres especiales:
  1. El `ESC (^[)` generado tecleando: `<ctrl><v>` seguido de `<esc>`
  2. El `RET (^M)` generado tecleando: `<ctrl><v>` seguido de `<ret>`
- Dos tipos de macros
  1. En modo comando
  2. En modo inserción

Los macros modo comando

- Se deben mandar llamar en modo comando
- Sintaxis general:
  `map <nombre-macro> <comandos> RET`
- Ejemplos macro modo comando:
  - `:map Q :q! ^M`
  - `:map d d4w ^M`
  - `:map v i \fB ^[ea \fP^[ ^M`

Los macros modo inserción

- Posible mandarlos llamar en modo inserción
- Sintaxis general:
  `map! <nombre-macro> <cadena-caracteres> RET`
- Ejemplos macro modo inserción:
  - `:map! ZZ ^[:wq ^M`
  - `:map! ^E \'{e} ^M`
  - `:map! ^[u \'{u} ^M`
  - `:map! ^[n \~{n} ^M`

Comentarios sobre macros

- Para borrar un macro es posible usar:
  `unmap`
- Posible definir comandos en archivo `.exrc`
- También posible definirlos en variable `EXINIT` de archivos `.profile` o `.login` (según sea el caso)
  `map <nombre-macro> <comandos> RET`
- En el caso de macros-inserción se tiene solo *un segundo* para introducir el nombre completo de la macro. En caso de que se tome más tiempo, el nombre del macro sera introducido como parte del texto editado.

Otros comandos de vi

Comando	Descripción
`J`	une la lineas actual con la siguiente
`.`	repite el más reciente comando
`:w` *arch*	escribe el buffer de trabajo en *arch*
`:r` *arch*	lee el contenido de *arch* y lo deja en el buffer de trabajo
`:q`	salir de vi
`:q!`	salir de vi sin guardar nada
`ZZ`	escribe buffer trabajo en el archivo actual y se sale de vi

`:f` ó `ctrl-G`	despliega el nombre del archivo, el status, el número de línea actual, número líneas en buffer y el porcentaje del buffer trabajo que precede a la línea actual
`G`	se va al final del archivo
`ctrl-v`	inserta el siguiente caracter literalmente
`~`	cambia minusculas por mayusculas y viceversa