Figura 1: Primer menú

```
Seleccione el sistema operativo con el que desea iniciar
   1.- Microsoft ...
   2.- Estación Linux
Use las flechas para resaltar la opción.
Luego presione Enter
```

1. Introducción.

Este documento es una guía para la realización de las prácticas de Programación de primer curso de I.T en Informática de la Universidad de Valladolid. No es nuestra intención, en absoluto, sustituir con él un buen libro sobre linux o Pascal.

2. Entrando en el sistema.

Encender la máquina

En primer lugar deberemos encender el ordenador y el monitor (los interruptores aparecen claramente identificados en el frontal del equipo ⏻).

Tras, posiblemente, una serie de mensajes de arranque, se verá una pantalla que muestra un menú como el recogido en la figura 1.

Hay que elegir la segunda opción: "Estación Linux" pulsando las flechas para que resalte y pulsando después la tecla [return] o [Enter] , (o bien esperando unos segundos).

Entrar en el sistema

Después, nuevamente, de una (larga) serie de mensajes (que reflejan el proceso de carga del sistema operativo), aparecerá una pantalla de saludo del sistema operativo (RedHat Linux) con una ventana en la que se ve el nombre de la máquina concreta (por ejemplo `l106a4`), el logotipo de la Escuela y dos casillas para escribir con las leyendas "**login:**" y "**password:**".

Es la pantalla de saludo de la máquina, en la que ésta se identifica ante nosotros y espera a que nosotros lo hagamos ante ella.

La identificación de un usuario ante el sistema se realiza mediante dos palabras: nombre de usuario («login») y contraseña («password»), que previamente nos habrá proporcionado el administrador del sistema.

Teclearemos entonces nuestro nombre de usuario, [return] y nuestra contraseña[1] [return] .

El nombre de usuario es nuestro nombre de acceso al sistema, y como tal es público, mientras que la contraseña es nuestra clave de acceso, que garantiza que sólo el usuario autorizado para ello puede utilizar un nombre de usuario. Por motivos evidentes la contraseña es privada.

Tanto nombre de usuario como contraseña son proporcionados al usuario por el administrador del sistema, y si olvidamos cualquiera de ellos deberemos consultar a éste.

Nombre de usuario y contraseña son, en resumen, la única garantía de que nadie nos suplanta frente al sistema, y de que los datos que almacenemos en él

[1] Por motivos de seguridad al teclear la contraseña no la vemos reflejada en pantalla

no serán utilizados, modificados o borrados por terceras personas, de modo que deben seguirse una serie de recomendaciones al respecto:

- No «prestar» la contraseña a nadie. La responsabilidad de *todo* lo que se hace desde una cuenta es del usuario de la cuenta.
- No anotar nunca la contraseña.
- En caso de anotar la contraseña, nunca dejar el papel a la vista, y mucho menos en la sala de ordenadores.

Tanto si el nombre de usuario es incorrecto como si el error se produce en la contraseña, el sistema contestará:

```
Login incorrect
```

y volverá a solicitar ambos datos. En caso contrario acabamos de entrar en el sistema y veremos una pantalla (gráfica) en la cual aparecen

- una barra inferior con una serie de «iconos» (dibujitos) de aplicaciones y utilidades
- dos «botones» en la parte superior izquierda que sirven para apagar y reiniciar el equipo

A partir de este momento estaremos trabajando en un «entorno» local, pero desde el que se accede a un sistema de almacenamiento de archivos en una máquina común a todos los alumnos (llamada jair). En ésta, cada alumno dispone de un espacio privado para almacenar sus archivos.

La máquina espera ahora que seleccionemos una orden con el "ratón", para ejecutarla inmediatamente.

La barra inferior muestra una serie de grupos de órdenes por categorías. Sobre algunas de ellas aparece un pequeño triángulo, sobre el que hay que pulsar para expandir el grupo y volver a pulsar para contraerlo. Además, una de las del grupo se considera más importante, y se puede ejecutar directamente pulsando sobre el icono del grupo.

En la parte central de la barra hay cuatro botones, nombrados *uno*, *dos*, *tres* y *cuatro*. Se refieren a cuatro posibles "escritorios" o lugares de trabajo, sencillamente para poder hacer varias cosas y tenerlas organizadas más cómodamente.

3. Saliendo del sistema

Abandonar el sistema

Al terminar la sesión de trabajo deberemos abandonar el sistema. La forma de hacerlo es pulsar sobre el botón ⏻ situado **en la pantalla** a la derecha de los escritorios numerados. Se pedirá confirmación para salir del sistema.

Apagar la máquina

Después habrá que apagar la máquina, pulsando sobre el botón *Apagar* **de la pantalla**, en la parte superior izquierda.

4. Escribiendo el primer programa.

Para ejecutar un programa Pascal primero debemos almacenarlo en el disco de jair y, para ello, debemos utilizar un *editor de textos*, en nuestro caso vi.

Figura 2: Fichero vacío

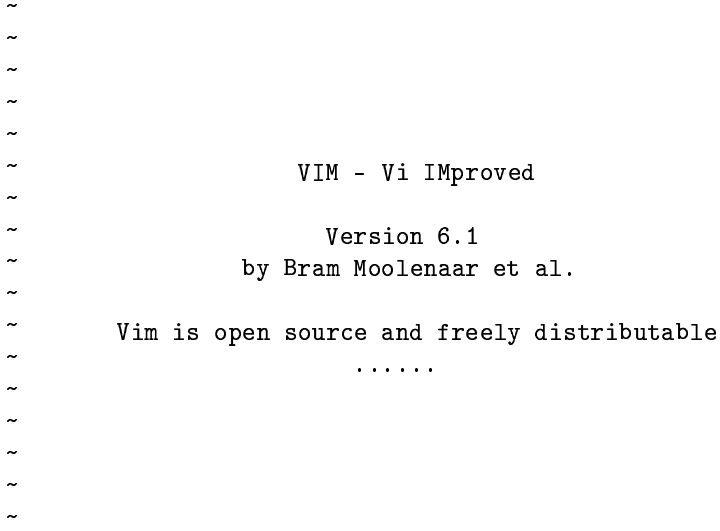


Figura 3: Primer programa en Pascal

```
program primero (output);

Begin
      writeln ('¡Hola!');
End.
```

vi es un editor de textos, que puede usarse para todo tipo de textos "planos", especialmente útil para la escritura de programas, incorporado a los sistemas UNIX™ y linux .

En el caso de este sistema, realmente se utilizará vim ("vi IMproved"), que que es una ampliación de vi con ciertas mejoras, o gvim (vim gráfico) que tiene algunas comodidades más.

Para activar gvim debemos pulsar sobre el icono que muestra un texto con un lapicero, de la barra inferior de tareas. Se obtendrá una ventana de edición, con algo como lo recogido en la figura 2. y una barra superior de grupos de órdenes y órdenes directas. Esto quiere decir que el editor gvim está activo y que podemos comenzar a escribir nuestro programa. Para ello tecleamos `i`, que no aparecerá en pantalla, y comenzamos a copiar «textualmente» el contenido de la figura 3 utilizando return para saltar de una línea a otra. Editar

El editor vi (y por lo tanto también vim y gvim[2]) tiene dos modos de trabajo: el modo "orden" (llamado también "modo comando"), en el que estamos al co-

[2]para resumir y aclarar ideas: gvim incorpora (usa) vim; vim es una extensión de vi

menzar la edición, y el modo de inserción (llamado "edición" en algunos textos), al que hemos pasado mediante el comando `i`. Al pulsar esc regresamos del modo de inserción al modo línea, con lo que podemos utilizar comandos como los recogidos en la sección 7 .

El modo de trabajo se muestra en la línea inferior de la pantalla del editor (nada si es modo orden, `INSERT` en modo inserción, etc). Esta línea se usa también para mostrar otras informaciones en modo orden.

vim y gvim reconocen también las teclas especiales de borrado, inserción etc. del teclado[3]

Grabar

Una vez terminada la edición, habrá que guardar el texto, con un nombre adecuado: por ejemplo `primero.pas` . Para ello, se pulsará esc seguido de `:w primero.pas` return (`w` de `write`). Como consecuencia, se habrá asignado el nombre al texto escrito, y podrá verse que gvim lo muestra en distintos colores, según la sintaxis de Pascal.

La extensión del nombre del fichero (lo que se escribe tras el punto, en nuestro caso, `pas`), es una forma de describir el tipo de archivo. Servirá para que diversos programas lo conozcan, el usuario sepa de un vistazo de qué se trata, además de permitir que vim detecte la sintaxis y la pueda mostrar. Si la extensión no es `pas` o `pp`, o el archivo aún no tiene nombre, vim no sabe que el texto está en Pascal, pero puede especificársele desde la barra superior en el grupo `Sintaxis`.

Terminar la edición

Para terminar la edición hay que pulsar esc seguido de `:q` return (`q` de `quit`, dejar la edición). Desaparecerá la ventana y el fichero habrá quedado almacenado. Si tratamos de abandonar la edición sin haber guardado el fichero, el editor lo advertirá. Se puede realizar la grabación y salir simultáneamente con la orden `:wq`, o abandonar la edición sin guardar la última versión con `:q!`

Todo este proceso puede realizarse también usando la barra superior, en el grupo `Archivo` órdenes `Guardar`, `Guardar como ...` etc.

Explorar el contenido del disco

Podemos comprobar si la edición del programa descrita en el apartado anterior ha sido correcta y el fichero ha quedado almacenado. Para ello hay que explorar el contenido del disco[4]. Hay varias maneras de hacerlo. Una es utilizar, de la barra inferior, el icono que parece una carpeta de cartón marrón: se obtendrá una ventana con el contenido del "directorio"[5].

Abrir una ventana "consola"

Otra forma es abrir una ventana "consola", que permite trabajar con el sistema operativo mediante teclado. Esto se consigue pulsando en el icono de la barra inferior que parece una pantalla con un $\surd$ en rojo. Entonces, la ventana que aparece es una vía de comunicación con el sistema operativo basada en orden-respuesta. El sistema escribe el nombre de usuario y el de la máquina y el directorio en el que se encuentra seguido de un símbolo $ ("prompt" del sistema, por ejemplo `[pepegon@l106a4 pepegon]$`). Está a la espera de una orden (comando). El usuario escribe una, seguida de return , y entonces el sistema actúa en consecuencia y pasa a estado de espera de la siguiente.

[3]Hay que señalar que el vi, en general, no reconoce estas teclas, salvo quizá las teclas de dirección (flechas, y sólo si estamos en el modo comando). En todo caso, en modo comando, los movimientos por el texto siempre pueden hacerse con las teclas `h`, `j`, `k` y `l` En modo inserción es posible que no se puedan hacer movimientos por el texto

[4]De la parte de disco que el usuario tiene asignada

[5]Conjunto de archivos del usuario, que se organiza jerárquicamente en "directorio principal" y "subdirectorios", repetidamente

Figura 4: Resultado de compilación

```
Free Pascal Compiler version 1.0.6 [2002/05/23] for i386
Copyright (c) 1993-2002 by Florian Klaempfl
target OS: Linux for i386
Compiling primero.pas
Assembling primero
Linking primero
4 lines compiled 0.2 sec
```

Desde una ventana como ésta, se pueden emitir todas las órdenes del sistema (a las que el usuario esté autorizado).

El comando

```
ls
```

ó bien

```
ls -l
```

da una lista de los ficheros que tenemos en el directorio actual. En este caso deberá aparecer un único elemento en la lista: el recién creado `primero.pas`.

Es recomendable tener siempre una ventana consola abierta para poder emitir órdenes al sistema cuando haga falta. Casi todo el proceso de desarrollo de programas lo necesita.

En particular, otra forma de acceder al editor es, desde una ventana consola, teclear

```
vim
```

(o `gvim`). Escribiendo

`vim` *«nombre».pas*

(o `gvim` <<*nombre*>>`.pas`), el editor arrancará cargando además el fichero cuyo nombre se especifica (y ahora ya se conocerá el tipo de fichero). Si no existía, se creará cuando se guarde.

5. Ejecutando el primer programa.

El programa `primero.pas` ya está almacenado, pero está en un lenguaje que el ordenador no entiende. Para ejecutarlo, es necesario previamente traducirlo a su lenguaje: "compilarlo".

Para compilar el fichero utilizaremos el comando Compilar

`fpc` *«fichero»*

que activa el compilador de Pascal: en nuestro caso `fpc primero.pas`.

El sistema irá mostrando el proceso hasta completar su tarea, como en la figura 4

Si obtenemos el resultado mostrado en la figura 4, y no algún mensaje de error como respuesta, habremos conseguido compilar correctamente nuestro primer programa. De lo contrario, habremos cometido algún error al teclear el comando o al crear el programa. En este caso deberemos volver a editar el programa como se describe en las secciones 4 y 7, bien con la orden `vim primero.pas` o desde el icono de `gvim` usando del grupo de órdenes `Archivo` la orden `Abrir`.

El comando `ls` permitirá comprobar que el compilador ha creado dos nuevos ficheros: `primero.o`, que contiene el código "objeto" de nuestro programa y el fichero `primero` que contiene ya el código ejecutable. Para ejecutar el programa teclearemos el nombre del fichero que lo contiene[6], en nuestro caso Ejecutar

```
./primero
```

y veremos en pantalla el resultado[7].

6. Más sobre linux

Insistimos en que este documento es sólo una introducción al sistema.

El sistema operativo linux dispone de multitud de comandos que permiten entre otras cosas manejar los ficheros del disco, además de gran cantidad de programas con utilidades diversas. En esta sección veremos algunos de los más significativos.

- Mostrar el contenido de un fichero. Si sólo queremos ver el contenido de un fichero no necesitamos editarlo con `vi`, basta con mostrarlo por pantalla con

  `cat` *«fichero»*

  El inconveniente de este método de trabajo es que todo el contenido del fichero pasará ante nuestros ojos a gran velocidad. Para detener la salida en cada página podemos redirigir la salida del comando a `more`, programa que "pagina" la entrada que recibe. Así

  ```
  cat primero.pas | more
  ```

  nos muestra `primero.pas` deteniéndose a cada página del fichero[8].

- Dividir el disco en trozos. El disco de jair está organizado en partes, denominadas directorios, que mantienen una estructura de árbol. El usuario, al entrar en el sistema, pasa a un directorio de su propiedad que podemos denominar mediante `$HOME`.

  Los alumnos de esta asignatura deberán utilizar un directorio denominado `PrEntregas` para la entrega de las prácticas. Para crearlo [9] utilizaremos el comando

[6] Ahora hay que dar el nombre completo, compuesto por el nombre que le dimos en la edición, precedido de su localización en la estructura de directorios. El `./` indica que se encuentra en el directorio establecido por defecto en este momento

[7] No es gran cosa, es cierto, pero sólo es el primer programa.

[8] Si, como es el caso, el fichero es pequeño, no notaremos los efectos de `more`, pero esta técnica será de gran utilidad más adelante. También es útil `less`, que hace lo mismo pero con más posibilidades, que se explican pusando la tecla `h` cuando se está usando

[9] Lógicamente el directorio debe ser creado una sola vez.

```
mkdir PrEntregas
```

Podemos pasar a ese directorio con la orden

```
cd PrEntregas
```

y volver a nuestro directorio raíz mediante

```
cd ..
```

o bien

```
cd $HOME
```

- Copiar un fichero a otra localización mediante

  `cp` *«fichero original» «directorio de destino»*

  ó bien

  `cp` *«fichero original» «fichero de destino»*

  En el primer caso hará una copia con el mismo nombre en la nueva localización. En el segundo, hará una copia con el nombre que se especificó como destino, que puede incluir una nueva localización.

- Cambiar de nombre un fichero mediante

  `mv` *«nombre original» «nuevo nombre»*

  (`mv` viene de "move", mover)

- Borrar un fichero mediante

  `rm` *«fichero»*

  (de "remove", borrar).

  Esta última operación es especialmente peligrosa puesto que *no podemos recuperar un fichero una vez borrado.*

  Confirmar sistemáticamente borrados

  Hay una forma de que el sistema pida confirmación siempre, antes de borrar definitivamente: consiste en decirle que, en lugar de referirnos a la orden `rm` a secas, nos queremos referir a la orden cualificada `rm -i`, que pide confirmación. Para ello, emítase la orden

  ```
  alias rm='rm -i'
  ```

  A partir de este momento, cada vez que escribamos `rm` el sistema ejecutará `rm -i` y podremos trabajar con algo más de seguridad.

  Como es posible que queramos actuar siempre así, y sería incómodo tener que establecer ese "alias" cada vez que entremos en el sistema, podemos escribir esta orden entre el conjunto de órdenes que se ejecutan siempre al entrar en él. Este conjunto de órdenes está almacenado en un fichero de nombre `.bashrc` y está en el directorio raíz. Forma parte de un grupo de ficheros que la orden `ls` no muestra, salvo que la cualifiquemos `ls -a`. Así que, abrimos con el editor el archivo `.bashrc`, y en él insertamos una línea

con el contenido `alias rm = 'rm -i'` y como consecuencia, a partir de ahora, cuando entremos en el sistema se establecerá la redenominación.

Algo parecido puede hacerse con `cp` y `mv`, o con las órdenes que nos parezca conveniente.

- Redirigir la salida de una orden. La salida de una orden puede "redirigirse" a otro lugar (fichero) añadiendo el cualificador `>` *« nombrefichero»*. Por ejemplo,

  ```
  ./primero >salida.txt
  ```

  no mostrará nada en pantalla, pero se habrá creado en el directorio un nuevo fichero, de nombre `salida.txt`, cuyo contenido es el que de otro modo se habría visto en la pantalla.

  Además, un doble >añadirá la salida al fichero de destino. Por ejemplo

  ```
  ./primero >> salida.txt
  ```

  hará que el fichero `salida.txt` contenga una línea más que antes.

- Obtener más información. El comando `man` nos ofrece información relativa a los comandos o programas instalados en el sistema. Así

  ```
  man ls
  ```

  nos brinda información sobre el funcionamiento y opciones de `ls` mientras

  ```
  man -k ls
  ```

  nos ofrece una pequeña descripción de todos los comandos relacionados con `ls`. En particular,

  ```
  man fpc
  ```

  muestra información sobre el compilador que usaremos en la asignatura.

- Repetir órdenes ejecutadas anteriormente: no es un estándar de los sistemas linux, pero en este sistema está definido que las flechas hacia arriba y hacia abajo permitan recorrer las órdenes tecleadas previamente, lo que ahorrará trabajo de escritura.

- Completar automáticamente órdenes y nombres: una orden a medio teclear, o el comienzo de un nombre de archivo, serán completados, si no hay ambigüedad, pulsando la tecla de tabulación $\boxed{\rightarrow|}$ Por ejemplo,

  `vi pri`$\boxed{\rightarrow|}$

  se convertirá en `vi primero` . Tampoco es un estándar de linux

7. Más sobre vi

El editor de textos vi dispone de muchos otros comandos que nos permiten modificar un fichero de texto previamente creado. En esta sección mostraremos sólo un resumen de algunos de ellos.

Todos los comandos deben ser activados desde el «modo comando». Como vimos anteriormente para pasar al modo comando debemos pulsar la tecla [esc] . Por supuesto si ya estamos en este modo no es necesario hacerlo, pero ello no produce ningún error, de modo que ante la duda es preferible teclear siempre [esc] seguido del comando o comandos que deseamos ejecutar.

Movimiento del cursor.

$\leftarrow \rightarrow \uparrow \downarrow$	Desplazar el cursor.
h l k j	Desplazar el cursor como las anteriores.
nG	Ir a la línea n.
$	Ir al final de la línea.
0	Ir al principio de la línea.
w	Ir al principio de la palabra siguiente.
[return]	Ir al principio de la línea siguiente.

Paso a modo INSERCIÓN de texto.

i	delante del cursor
a	tras el cursor
A	a continuación del final de la línea actual
O	en una nueva línea que crea antes de la actual
o	en una nueva línea que crea detrás de la actual

Borrado de texto.

x	Borra el carácter bajo el cursor (y lo almacena en su memoria)
dd	Borra la linea bajo el cursor (y lo almacena en su memoria)
dw	Borra la palabra bajo el cursor (y lo almacena en su memoria)

Copiado de texto.

p	Copia el contenido de su memoria detrás de la posición actual.
P	Copia el contenido de su memoria delante de la posición actual.
yy	Copia la línea en su memoria.
$n$yy	Copia n líneas en su memoria.

Salida del editor.

:w [return]	Escribe en el disco el contenido actual del fichero
:q [return]	Abandona el editor (solicitando confirmación de grabado)
:q! [return]	Abandona el editor (fuerza al editor a salir sin grabar)
:wq [return]	Abandona el editor tras grabar

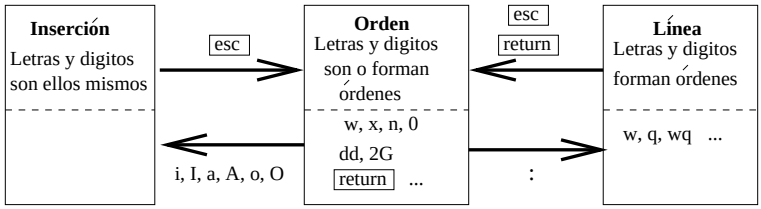
Otros comandos.

<Ctrl>L	Refresca la pantalla.
<Ctrl>G	Informa sobre el número de linea actual y nombre del fichero.
/*texto* [return]	Se coloca sobre la siguiente aparición de *texto*.
n	Se coloca sobre la siguiente aparición de lo último buscado (con /)
J	Funde la línea actual con la siguiente

Gran parte de los comandos de `vi` admiten la repetición. Así por ejemplo si deseamos borrar 10 caracteres el comando se escribe como `10x`. Esto puede crear problemas si, por ejemplo, tecleamos inadvertidamente un número , digamos 100, seguido del comando `i` y procedemos a insertar un texto en el fichero. Al salir del «modo de inserción» ¡todo el proceso se repetirá 100 veces!

Hay un gran grupo de comandos, aparte de los descritos, que comienzan con el carácter : Para todos ellos se ve en la última línea la composición de la orden, de forma que podría considerarse que hay un tercer modo de edición: el modo "línea", al que se accede desde el modo orden con el comando : y del que se sale a modo orden con esc (o una vez completada la orden de línea).

Figura 5: Modos de edición



Hay muchísimas más órdenes. Se puede acceder a una ayuda mediante el comando `:help` return (en inglés, claro).

8. Desarrollo de programas y errores

Se basa en el esquema de la figura 6.

La orden `fpc` realiza las fases de compilación propiamente dicha, ensamblado y enlace.

La **compilación** (propiamente dicha) se encarga de traducir el programa Pascala código *ensamblador*.

El **ensamblado** traduce el código ensamblador obtenido a código máquina ("*relocalizable*").

El **enlace** combina el código máquina relocalizable con las rutinas necesarias (que estarán almacenadas en las llamadas *bibliotecas* o "library"), y resuelve otros problemas, para obtener finalmente el código máquina ejecutable.

En cualquiera de los procesos de análisis, diseño, y codificación (traducción a Pascal) pueden producirse errores. Los cometidos en la edición como pulsaciones incorrectas de teclas etc. son responsabilidad del editor, y se advierten durante la edición.

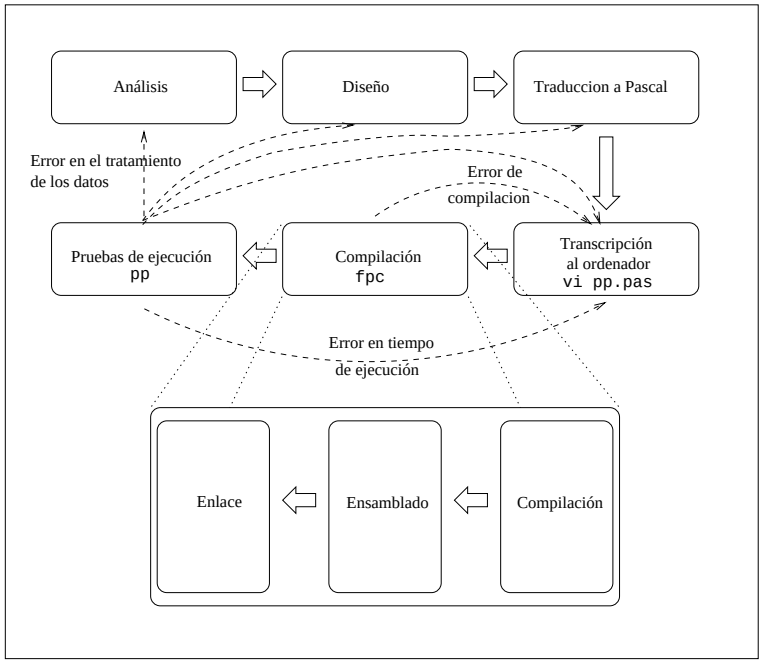
Los errores de otro tipo se detectan en la fase de compilación o de ejecución.

Cuando el compilador detecta un error en alguna de sus fases, informará del mismo, de forma que, si es importante, no se podrá generar la salida deseada.

`fpc` considera varios tipos de errores, de los que informará según su gravedad ("warning", "error", "fatal error" ...)

Con frecuencia, el proceso de depuración, es decir, localización, interpretación y corrección de errores, es complicado. A veces, un error que se produce en un determinado lugar del código, o del proceso, no se detecta hasta un lugar o momento posterior.

Figura 6: Desarrollo de programas



También es posible que un error, quizá simple, ocasione una cascada de detección de errores posteriores, que realmente no lo son cuando se corrige el que los originó.

Por otra parte, los compiladores no pueden detectar todos los errores, en particular si son de análisis o diseño. Algunos de estos se detectarán en la ejecución, y otros tampoco así. Por ello es importante disponer de una adecuada batería de pruebas.

Con programas algo mayores, es frecuente que el informe de errores de la compilación exceda el tamaño de la pantalla. En este caso, habrá que redireccionar (>) o filtrar (`| more`), no la salida del compilador, sino la salida de error, que se envía al "dispositivo" 2, por ejemplo con la orden

```
fpc pp.pas 2>errores.txt
```

que generará un nuevo fichero, de nombre `errores.txt` con toda la información, y se podrá estudiar con detalle.

Puede ser útil insertar en el código sentencias que muestren los valores de las variables en momentos significativos para poder seguir el algoritmo en sus partes críticas, y eliminarlas cuando el programa esté depurado.

Para la búsqueda de errores detectados en compilación existen algunas reglas muy útiles:

1. Leer el mensaje de error
2. Leer el mensaje de error (esta línea no es un error)

3. No modificar nunca el programa sin reflexionar sobre las causas del error, ni ensayar soluciones arbitrarias.

Si a pesar de todas las precauciones, provocamos una ejecución en la que se entre en un "bucle infinito", la pulsación de la combinación de teclas Control y C detendrán dicha ejecución. Si aún así no se consigue, todavía se puede detener mediante

`kill -9` *número de proceso*

El número de proceso lo muestra la orden

`ps -u` *nombre de usuario*

9. Después de la primera sesión

El objetivo de las prácticas no es el aprendizaje del sistema operativo linux, pero es evidente que el manejo con soltura del sistema nos permitirá resolver las prácticas de un modo más rápido y cómodo, tanto en ésta como en otras asignaturas.

En todo caso, nada podrá proporcionar conocimientos suficientes como el estudio y la práctica personal, ayudada por la consulta a la amplia bibliografía existente sobre estos temas. En este sentido, para quien pueda y quiera practicar con una máquina en casa, es posible descargar el compilador `fpc` desde la página

`http://www.freepascal.org`

para plataformas DOS, linux, Windows™ y otras.

Se recomienda una visita de todas formas a esta página, donde además se encontrará abundante información sobre `fpc`.

Para acceder a los datos almacenados en jair se puede usar el programa `ssh`, que puede conseguirse desde las páginas del laboratorio

`http://www.lab.fi.uva.es`