Unitat Formativa 3. Gestió de Fitxers

1.1. Concepte i tipus de fitxers

El fitxer és el component bàsic de qualsevol sistema informàtic pel tractament de dades. Un dels problemes amb el que ens hem trobat fins ara és que l'emmagatzematge de dades en variables i vectors és temporal, al terminar el programa les dades es perden. Per a conservar permanentment la informació hem d'utilitzar fitxers, que ubicarem a unitats de disc o xarxa.

També permet descarregar la memòria del sistema de les dades que ja no son utilitzades, així quan un programa acaba amb informació que no serà necessària posteriorment, la guarda en un fitxer alliberant la memòria que ocupava la informació, retornant-li memòria al sistema.

Estructura de la informació:

- Una dada (camp) és un conjunt de bits combinats per formar caràcters o bytes amb significat propi.
- Un *registre* (que es pot representar com una estructura de dades complexes en forma de vector associatiu) és un conjunt de camps que tenen relació entre sí (com els camps d'un DNI).
- Un fitxer és un conjunt de registres relacionats entre sí i que podrien estar ordenats en funció d'un camp considerat com ***camp clau***. El fitxer peça bàsica per a la construcció de la majoria dels S.I. i S.O.

Tipus de fitxers

Segons el seu contingut	
Executables	Programes o aplicacions que realitzen tasques sobre el S.O.
Codi	Text font d'un programa executable
Dades	Informació útil per a el funcionament dels programes,el S.O. o l'empresa
Binaris	Es llegeixen i guarden tal i com procedeixen d'un dispositiu físic sense realitzar cap conversió
Segons el seu temps d'existència	
Permanents	Existeixen pràcticament sempre (dades de l'empresa, paràmetres d'inici d'un programa)
Temporals	Només existeixen durant l'execució d'un programa i es solen eliminar quan finalitza.
Segons la seva organització interna o forma de accedir al contingut	
Seqüencial	Les dades es guarden de forma contigua, s'aprofita més l'espai però és mes lent trobar una determinada dada i no és pot eliminar o modificar registres sense reescriure tot el fitxer, per que cada registre pot tenir una mida diferent.
Indexada	Tots els registres tenen la mateixa mida encara que no s'aprofiti al 100% i es guarden de forma ordenada per un camp anomenat clau. És la organització més idònia per localitzar i actualitzar registres.

ORGANITZACIÓ INTERNA DELS FITXERS

Organització Seqüencial

Es el mètode més simple, les dades es guarden de forma consecutiva i els registres o línees del fitxer poden tenir diferent mida. Les dades d'aquest tipus d'organització poden no estar ordenades.

Avantatges:

- Només es "gasten" els bytes estrictament necessaris al dispositiu escollit (optimització de l'espai).
- Per afegir dades només cal escriure a partir de la última posició del fitxer

Inconvenients:

- Per accedir a una determinada línea o registre cal llegir totes les anteriors (sistema lent)
- No poden inserir o modificar dades dels registres sense reescriure el fitxer.

Organització Indexada o de Accés Directe

Tots els registres (línies) del fitxer són de longitud fixa (independentment de la mida realment necessària per a les dades de cada línia), i un dels camps del registre ha de tenir un valor **Únic** (camp clau que acostuma a ser un número de línia).

Els registres han de estar ordenats pel valor d'aquest camp, per permetre l'**accés directe** a un registre específic, ja que només cal multiplicar el valor del camp clau per la longitud fixa del registre i podrem conèixer la seva posició exacta dins del fitxer.

Avantatges

- L'accés a les dades és molt més ràpid
- Podem accedir a les dades tant de forma seqüencial com directament segons ens interessi.
- Es poden esborrar i modificar dades i registres sense reescriure tot el fitxer

Inconvenients

- El fitxer tindrà major mida que si fos de tipus seqüencial.
- Els valors del camp clau han de ser únics i sense deixar "forats", per exemple evitar que del valor 7 passem al 11, caldrà mantenir els espais intermedis encara que els registres restin buits o bé, si es possible fer-ho, reorganitzar i reescriure els valors del camp clau.

1.2. Operacions sobre fitxers seqüencials i indexats.

Les operacions típiques sobre fitxers son: creació, apertura/tancament i lectura/escriptura. Podem afegir matisos a la escriptura realitzant una operacions com afegir, i en cas de fitxers indexats podem actualitzar dades llegint i escrivint a una posició determinada dins del fitxer (posicionament).

Referències: https://www.php.net/manual/es/ref.filesystem.php

fopen() *https://www.php.net/manual/es/function.fopen.php*

Permet crear i obrir un fitxer. Aquesta funció, si té èxit, ens retorna un apuntador que ens connecta a un fitxer del sistema operatiu, en cas contrari ens retorna un **false**.

Una vegada obtingut el apuntador podem realitzar a través d'ell les operacions de lectura, escriptura o posicionament sobre el fitxer físic a partir d'un conjunt de funcions. Com es pot veure a la documentació oficial, pot rebre un conjunt de paràmetres, però els principals son el **filename** o "ruta+nom" del fitxer a tractar i el mode d'accés al fitxer.

El **filename** pot ser una URL o un recurs local. Cal destacar que amb Sistemes Windows el caràcter especial barra invertida **** ens obliga a "escapar" dels seus efectes anul·lant-los amb doble barra invertida \\ , i sempre que sigui possible fer servir la barra **/** típica dels Sistemes Unix/Linux.

El **mode de accés** ens permet realitzar unes accions determinades sobre el fitxer:

mode	Descripción
'r'	Apertura para sólo lectura; coloca el puntero al fichero al principio del fichero.
'r+'	Apertura para lectura y escritura; coloca el puntero al fichero al principio del fichero.
'w'	Apertura para sólo escritura; coloca el puntero al fichero al principio del fichero y trunca el fichero a longitud cero. Si el fichero no existe se intenta crear.
'w+'	Apertura para lectura y escritura; coloca el puntero al fichero al principio del fichero y trunca el fichero a longitud cero. Si el fichero no existe se intenta crear.
'a'	Apertura para sólo escritura; coloca el puntero del fichero al final del mismo. Si el fichero no existe, se intenta crear. En este modo, fseek() solamente afecta a la posición de lectura; las lecturas siempre son pospuestas.
'a+'	Apertura para lectura y escritura; coloca el puntero del fichero al final del mismo. Si el fichero no existe, se intenta crear. En este modo, fseek() no tiene efecto, las escrituras siempre son pospuestas.
'x'	Creación y apertura para sólo escritura; coloca el puntero del fichero al principio del mismo. Si el fichero ya existe, la llamada a **fopen()** fallará devolviendo **FALSE** y generando un error de nivel **E_WARNING**. Si el fichero no existe se intenta crear. Esto es equivalente a especificar las banderas O_EXCL|O_CREAT para la llamada al sistema de open(2) subyacente.
'x+'	Creación y apertura para lectura y escritura; de otro modo tiene el mismo comportamiento que 'x'.

mode	Descripción
'c'	Abrir el fichero para sólo escritura. Si el fichero no existe, se crea. Si existe no es truncado (a diferencia de 'w'), ni la llamada a esta función falla (como en el caso con 'x'). El puntero al fichero se posiciona en el principio del fichero. Esto puede ser útil si se desea obtener un bloqueo asistido (véase flock()) antes de intentar modificar el fichero, ya que al usar 'w' se podría truncar el fichero antes de haber obtenido el bloqueo (si se desea truncar el fichero, se puede usar ftruncate() después de solicitar el bloqueo).
'c+'	Abrir el fichero para lectura y escritura; de otro modo tiene el mismo comportamiento que 'c'.
'e'	Establecer la bandera 'close-on-exec' en el descriptor de fichero abierto. Disponible solamente en PHP compilado en sistemas que se ajustan a POSIX.1-2008.

Al mode d'accés encara se li pot afegir al final una lletra 't' o 'b', indicant si volem accedir a les dades en mode text o binari.

Ho trobareu explicat amb detall amb l'enllaç però en resum: "encara que treballant amb Sistemes Windows ens resultarà pràctic accedir amb la 't' pels finals de línia, hem de mirar per la portabilitat del nostre codi i treballar amb la 'b' per accedir en mode binari."

```
$file = fopen ('d:\\fitxers\\php\\project3\\users.txt', 'rb');
```

fclose() https://www.php.net/manual/es/function.fclose.php

Tanca el fitxer assegurant la salvaguarda de totes les modificacions. El seu ús és dels més simples que podem trobar, simplement rep per paràmetre el apuntador al fitxer que volem tancar. Retornant un true o un false en funció del seu èxit.

```
fclose ($file);
```

file_exists() https://www.php.net/manual/es/function.file-exists.php

Comprova si el fitxer ja existeix.Rep el filename del fitxer o directori, retornant true si existeix o false en cas contrari.

```
if ( file_exists ('d:\\fitxers\\php\\project3\\users.txt') == false ) {
        echo 'El fitxer no existeix';
}
```

fgets() https://www.php.net/manual/es/function.fgets.php

Permet llegir el contingut del fitxer línia a línia a partir de la posició on a quedat l'apuntador, que s'actualitzarà després de cada lectura per poder llegir el fitxer fins al final. Rep com a paràmetre l'apuntador a fitxer, i opcionalment, el nombre de caràcters a llegir, sinó s'especifica quants caràcters es volen llegir s'aturarà al final de línia o bé al final de fitxer. Retorna la línia llegida, o bé false indicant que ja ha arribat al final del fitxer.

```
while ( ($line = fgets($file) ) !== false) {
        echo $line . "<br>";
}
```

fwrite() https://www.php.net/manual/es/function.fwrite.php

Igual que el seu alies **fputs**, permeten escriure al fitxer a partir del apuntador. Rep com a paràmetres l'apuntador a fitxer, la cadena o string a guardar al fitxer, i opcionalment, el nombre de caràcters del string a escriure al fitxer. Retornant el nombre de bytes escrits o false en cas de error.

```
fwrite ($file, "Arthur");
```

file() https://www.php.net/manual/es/function.file.php

Transfereix el contingut del fitxer a un array, omplint una posició per cada línia del fitxer. Rep com a paràmetre un string amb el pathname del fitxer a transferir.

```
$lines = file('d:\\fitxers\\php\\project3\\users.txt');

foreach ($lines as $linenumber => $line) {
   echo "Línia [$linenumber]: $line<br>";
}
```

file_get_contents() https://www.php.net/manual/es/function.file-get-contents.php

Transfereix el contingut del fitxer a una cadena. Rep com a paràmetre un string amb el pathname del fitxer a transferir.

```
$datafile = file_get_contents('d:\\fitxers\\php\\project3\\users.txt');
```

Exemple complet:

```
if ( file_exists('d:\\fitxers\\php\\project3\\users.txt') == false ) {
   if ( ($file = fopen ('d:\\fitxers\\php\\project3\\users.txt', 'wb'))  != false ) {
      fwrite($file, "Arthur\n");
      fwrite($file, "Richard\n");
      fwrite($file, "Jane");
      fclose($file);
      $lines = file('d:\\fitxers\\php\\project3\\users.txt');
      foreach ($lines as $linenumber => $line) {
         echo "Línia [$linenumber]: $line<br>";
      }
   }
} else {
   if ( ($file = fopen ('d:\\fitxers\\php\\project3\\users.txt', 'rb')) ) {
      while ( ( $line = fgets($file) ) !== false ) {
         echo $line . "<br>";
      }
      fclose($file);
      $datafile = file_get_contents('d:\\fitxers\\php\\project3\\users.txt');
      echo $datafile;
   }
}
```

Per a fitxers indexats on totes les línies tenen la mateixa mida podem fer servir les següents funcions:

fread()

https://www.php.net/manual/es/function.fread.php

Llegeix una quantitat establerta de caràcters Rep com a paràmetres l'apuntador a fitxer i un sencer que estableix el nombre de caràcters a llegir. Retornant la cadena llegida o false en cas de error.

```
while ( ($line = fread($file, 80) ) !== false) {
        echo $line;
}
```

I les funcions sobre el posicionament intern de l'apuntador sobre el fitxer:

ftell() Retorna a quina posició del fitxer apunta actualment l'apuntador
$pos = ftell ($file);

fseek() Estableix la posició de l'apuntador dins del fitxer
fseek ($file, 240, SEEK_SET); //avança 3 línies

rewind() Retorna l'apuntador a la posició inicial sobre el fitxer
rewind ($file);

1.3. Modularització de les operacions sobre fitxers.

Es fonamental realitzar les tasques més habituals utilitzant funcions, per aconseguir una major productivitat i aprofitament del codi que ja ha estat desenvolupat. El concepte implícit es reaprofitar el codi genèric que ja està provat i és perfectament funcional, i no treure còpies d'un codi específic i de gran mida, i adaptar-les a cada cas particular.

Així podem modularitzar accions genèriques repetitives com la de llegir tot el contingut d'un fitxer i guardar-lo a un array, o la de reescriure un fitxer a partir d'un array:

```
function save_data (string $pathname, array $data) : int {
   $file = fopen($pathname, 'wb');
   if ($file) {
      $registers = 0;
      foreach ($data as $line) {
         $line[strlen($line) - 1] = "\n";
         fwrite($file, $line);
         $registers++;
      }
      fclose($file);
      return $registers;
   } else {
      return -1;
   }
}
```

```
function read_data (string $pathname, array &$data) : int {
      if ( ($file = fopen($pathname, 'rb')) ) {
         $registers = 0;
         while (($line = fgets($file) ) !== false) {
            $data[] = $line;
            $registers++;
         }
         fclose($file);
         return $registers;
      } else {
         return -1;
      }
}
```

NOTA: Els codis retornats han de seguir una lògica. Així un valor positiu indica el nombre de registres tractes amb èxit. Un cero indica que no hem tractat cap registre. I cada valor negatiu indica un tipus de errada específic durant el tractament de les dades.

Enviament de variables a una funció

No sempre es suficient treballar amb variables de tipus primitius (**int, float, string** o inclús **array** que no és primitiu), per algunes tasques cal enviar dades complexes a través dels **arrays associatius**, unes estructures de dades que són un conjunt de parelles clau-valor amb les que treballarem molt sovint, normalment sota codificacions JSON https://www.json.org/json-es.html

Els arrays associatius es passen exactament igual que qualsevol variable a una funció, únicament hem de recordar que, si volen modificar el seu contingut, hem de definir a la funció que aquesta variable es rebrà per referència amb l'operador **&**.

Exemple d'ús:

Si treballem amb funcions que gestionen fitxers que contenen dades sobre els usuaris del nostre portal, el tractament d'aquestes dades haurà de ser específic a aquest cas. Això obliga a dissenyar funcionalitats específiques, que poden reutilitzar les funcionalitats genèriques ja desenvolupades, per adaptar les dades a guardar i recuperar, així com el controlador d'aquestes operacions.

Un exemple de funció específica seria una que prepara les dades del array associatiu per donar-li el format que necessitem abans de guardar-los al fitxer:

```
function usersToFileAdapter ( array $data ) : array {
   $registers = [];
   foreach ( $data as $user )  {
      $line = "";
      foreach ( $user as $valor )  {
         $line = $line . $valor . ",";
      }
      $line[strlen($line) - 1] = " ";
      $line = trim($line);
      $registers[] = $line ;
   }
   return $registers;
}
```

Un altre seria una funció que, a partir del array que hem obtingut de llegir el fitxer, ens prepari l'array associatiu amb el que treballarà la resta del projecte:

```
function usersFromFileAdapter ( array $registers ) : array {
   $data = [];
   for ($i = 0; $i < count($registers); $i++) {
      $registerdata = explode(",", $registers[$i]);
      $data[] = ["player"=>$registerdata[0], "level"=>$registerdata[1], "points"=>$registerdata[2]];
   }
   return $data;
}
```

Finalment, tindrem un controlador per gestionar les accions de lectura i escriptura de dades sobre fitxers. A mode de exemple podem provar les funcionalitats amb un controlador de proves com el següent:

DemoUsersFileController.php

```
//una variable rebrà la acció que li demanen al controlador, per exemple des d'un formulari
$operation = filter_input ( ... );
// i el pathname del fitxer que cal gestionari
$pathname = 'd:\\fitxers\\php\\users.txt';

// array associatiu amb les dades dels nostres usuaris
$users = [
   ["user" => "jose", "level" => 10, "points" => 1000],
   ["user" => "jordi", "level" => 1, "points" => 100],
   ["user" => "joan", "level" => 2, "points" => 200]
];

// provem de guardar el array associatiu amb les dades dels nostres usuaris
$operation = "w";
$data = [];

if ($operation == "w") {
   $data  = usersToFileAdapter($users);      // adaptem les dades
   var_dump($data);                          // es mostra el resultat
   $result = save_data($pathname, $data);  // salvem les dades al fitxer
   var_dump($result);                        // es mostra el nombre de registres guardats
}

// provem de llegir un fitxer que conté dades dels nostres usuaris i obtenir un  array associatiu
$operation = "r";
$rawdata = [];

if ($operation == "r") {
   if (($result = read_data($pathname, $rawdata)) > 0) {    // llegim les dades del fitxer
      var_dump($rawdata);                                   // es mostrem les dades
      $data = usersFromFileAdapter($rawdata);               // es construeix el array associatiu
      var_dump($data);
   }
   var_dump($result);                  // es mostra el nombre de registres llegits
}
```