L'intèrpret PHP

L'intèrpret PHP (http://www.php.net) és un programa (lliure i multiplataforma) que s'encarrega d'executar scripts escrits en el llenguatge PHP (http://php.net/manual/es). Aquests scripts poden fer pràcticament de tot: des de llegir o escriure fitxers fins accedir a bases de dades, passant per gestionar imatges o documents PDFs o xifrar/desxifrar dades, entre moltes altres coses més. En l'entorn web (que és on el llenguatge PHP se sol fer servir més i és el context on l'estudiarem) permet generar contingut web dinàmic, rebre valors de formularis, gestionar sessions d'usuari i cookies, etc.

Esquema general d'ús de l'intèrpret PHP via servidor HTTP

El més habitual és que el codi PHP s'escrigui embegut dins de codi HTML/CSS/Javascript (el qual és imprescindible per formar una pàgina web bàsica) per així ampliar enormement les capacitats i funcionalitat de les pàgines webs en qüestió, convertint-les llavors, de fet, en totes unes aplicacions senceres capaces de rebre dades interactivament de l'usuari, d'accedir a fitxers o a bases de dades, d'executar comandes, etc.

> NOTA: Grans gestors de contingut com Wordpress, Drupal o altres plataformes web com Nextcloud, Moodle o MediaWiki estàn desenvolupats amb PHP.

El procés d'"executar" una pàgina HTML/CSS/Javascript+PHP (a partir d'ara, per simplificar, direm "pàgina PHP" amb el benentès de que no es pot escriure una pàgina web sense una mínima estructura HTML obligatòria) és, simplificant molt, així:

1) El client HTTP (un navegador generalment) sol.licita una pàgina PHP (a través de la barra de direccions, per exemple) a un determinat servidor HTTP (Apache, per exemple)

2) Qualsevol programa servidor HTTP (Apache, Nginx, etc) per sí sol només és capaç de retornar directament pàgines que només continguin codi HTML, CSS i/o Javascript. Per tant, en carregar i detectar que la pàgina demanada incorpora codi PHP, aquest servidor no pot entregar-la directament sinó que li hauria de traspassar (si està ben configurat) tot el codi d'aquesta pàgina a l'intèrpret PHP

> NOTA: Les tecnologies HTTP, CSS i Javascript són, de fet, com ja sabem, les úniques tecnologies amb què el codi font d'una pàgina web pot estar escrit per tal que els navegadors el puguin entendre i, per tant, "renderitzar" el seu resultat visible a pantalla. Per tant, tampoc tindria sentit que els servidors webs puguessin enviar pàgines PHP directament als navegadors perquè aquests no entendrien el codi corresponent

3) L'intèrpret PHP processa tot el codi PHP que hi hagi en aquesta pàgina, obtenint com a resultat una pàgina HTML pura (deixant intacte el possible codi CSS i/o Javascript que hi pugui tenir). Per exemple,si inicialment la pàgina conté una línia PHP tal com *print Date("Y/m/d");* , l'intèrpret la substituirà per la cadena "2022/11/23" (o la data que sigui). Cal dir que les línies PHP poden realitzar molts tipus d'accions diverses, com ara l'accés a bases de dades, on la pàgina HTML pura retornada contindria en aquest cas les dades concretes obtingudes després d'haver realitzat la consulta pertinent en aquell moment. En qualsevol cas, la pàgina HTML pura serà retornada per l'intèrpret PHP al servidor HTTP

4) El servidor HTTP ara ja podrà retornar al client la pàgina HTML amb el resultat d'haver executat el codi PHP que tenia inicialment.

La gran diferència entre el llenguatge PHP i el llenguatge Javascript és que aquest últim és un llenguatge "client"; és a dir, són els navegadors els qui interpreten codi Javascript (a l'igual que codi HTML i/o CSS) directa i autosuficientment. En canvi, el llenguatge PHP és un llenguatge "servidor" perquè només l'intèrpret PHP (invocat per un servidor HTTP) és capaç d'entendre codi escrit en aquest llenguatge: com hem dit, si un navegador per si sol intentés obrir una pàgina PHP no funcionaria correctament perquè no pot entendre-la. És per això que per poder visualitzar correctament una pàgina PHP cal realitzar tot el procés descrit anteriorment de "masticació" del codi PHP en codi HTML, procés on intervé, com hem explicat, un servidor web funcionant com "intermediari" entre les peticions rebudes per part dels clients i el retorn a aquests del resultat obtingut de l'intèrpret PHP.

En aquest sentit, per realitzar proves ràpides sense que haguem d'instal.lar tot un servidor HTTP complet podem visualitzar pàgines PHP a través del navegador gràcies a un miniservidor HTTP que el propi intèrpret PHP incorpora "incrustat" dins seu. Per posar en marxa aquest miniservidor oferint una determinada pàgina PHP (l'anomenarem "pagina.php", amb un contingut PHP qualsevol, com per exemple el següent), *<html><body><?php phpinfo(); ?> </body></html>*" , només cal executar la següent comanda: ***php -S 0.0.0.0:8000 pagina.php*** (on el valor que acompanya el paràmetre *-S* indica la IP i el port per on escoltarà les peticions aquest miniservidor; si s'escriu la IP 0.0.0.0 vol dir que estarà escoltant per totes les IPs existents al sistema; també podria indicar-se la IP 127.0.0.1 si només es vol utilitzar el miniservidor en local)

NOTA: Si no s'indica cap nom de pàgina PHP, el miniservidor intentarà oferir una anomenada "index.php" o, en cas de no trobar-la, anomenada "index.html". En qualsevol cas, la pàgina PHP s'anirà a buscar a partir de la carpeta des d'on s'hagi executat la comanda *php*. Si es vol buscar en una altra carpeta concreta, es pot indicar afegint el paràmetre ***-t /carpeta/on/es/troben/els/scripts***

Maneres d'executar l'intèrpret PHP dins d'un servidor web

Hi ha diferents maneres de realitzar el punt 2) de la llista anterior (és a dir, de cridar, des del servidor HTTP, l'intèrpret PHP). Depenent dels seus "pros" i "contres", ens interessarà més una forma o una altra; i segons la decisió que prenem, la configuració del servidor web serà diferent. Bàsicament, hi ha tres opcions:

A) Cridar a l'intèrpret PHP funcionant com un programa independent de l'executable Apache (mode "CGI"). Això significa que cada cop que s'ha de realitzar el punt 2), l'Apache crida a un executable PHP extern i aquest es posa en marxa sempre com si fos la primera vegada (rellegint un altre cop la seva configuració particular, etc.). L'intèrpret PHP s'executarà amb uns permisos propis diferents dels de l'Apache i, de fet, fins i tot podria funcionar en una màquina separada diferent.

B) Cridar a l'intèrpret PHP funcionant com una llibreria dinàmica interna (en Windows, una "dll"; en Linux una "so") pertanyent al propi executable Apache (mode "mòdul"). Això significa que l'intèrpret PHP forma part del propi servidor web i, per tant, es posa en marxa en iniciar-se l'Apache (llegint la seva configuració en aquell moment només) i es manté llest mentre l'Apache està funcionant sense la necessitat de cap executable extern. De fet, l'intèrpret PHP s'executarà amb els permisos propis de l'Apache.

C) Darrerament s'està implementant una tercera manera de cridar a l'intèrpret PHP anomenada "FastCGI" (concretament el subtipus "PHP-FPM", de "FastCGI Process Manager") el qual és semblant en concepte al "CGI" però molt més ràpid ja que internament utilitza uns sistemes de comunicació entre el servidor Apache i l'intèrpret PHP diferents del mode "CGI" clàssic. Una altra diferència és que en aquest mode l'intèrpret PHP funciona com un servei/dimoni del sistema (en comptes d'un executable estàndar com pasava al mode "CGI" tradicional) que es manté contínuament en marxa escoltant les possibles peticions que provinguin en qualsevol moment de l'Apache (i això permet no repetir tasques preparatòries a cada petició com pasava amb el mode "CGI" tradicional).

L'avantatge més obvi del mode "mòdul" respecte el FastCGI és la facilitat de guardar l'"estat de la sessió" (és a dir, compartir dades comunes) entre dues peticions, cosa que amb els modes (Fast)CGI és possible però és més difícil perquè en ells cada petició és "la primera". D'altra banda, l'inconvenient més important del mode "mòdul" és que només es pot fer servir amb el worker "prefork"; a més, en aquest mode el procés Apache incorpora una "motxilla" PHP que demana gran quantitat de memòria i capacitat de processament del sistema, independentment de si les pàgines demanades són PHP (dinàmiques) o HTML/CSS/Javascript pures (estàtiques) i això fa, a la pràctica, que el rendiment del servidor sigui pitjor (és a dir, no sigui capaç de respondre a les múltiples peticions concurrents amb velocitat) D'altra banda, en el mode "mòdul" l'intèrpret PHP s'executa sempre amb el mateix usuari que el servidor Apache ("www-data" a Ubuntu, "apache" a Fedora) però en el mode FastCGI el servidor PHP-FPM es pot configurar per ser executat amb qualsevol usuari que es desitgi (o bé un de propi pel servidor en sí o fins i tot un de diferent per cada "virtual host"), millorant així la seguretat del sistema. És per tot això que, a la pràctica, el mode FastCGI ofereix un rendiment molt millor que el mode "mòdul", així que nosaltres aquí estudiarem el primer.

NOTA: "FPM" és una manera de dir que l'intèrpret PHP s'executa com un servidor que només és responsable de rebre les peticions de l'Apache i de gestionar en conseqüència diversos processos PHP fills (és a dir, crear-ne, matar-ne, etc) per tal de reenviar-li cada petició rebuda a un d'aquests processos fills en concret, el qual serà qui s'encarregarà realment de processar la petició i obtenir la resposta, la qual serà retornada al servei PHP-FPM i aquest la reenviarà a l'Apache.

NOTA: A https://blog.ahierro.es/php-mod_php-vs-cgi-vs-fastcgi-vs-fpm/ teniu un article comparant més exhaustivament les característiques de les diverses formes de comunicació entre un servidor web i un intèrpret PHP (via mòdul, via CGI, via FastCGI/PHP-FPM, etc)

Instal.lació de l'intèrpret PHP en mode "FastCGI" (PHP-FPM)

Podrem fer servir l'intèrpret PHP via FastCGI si instal.lem el paquet "**php-fpm**" (el qual utilitza el mòdul "*proxy_fcgi*" de l'Apache per funcionar, però aquest ja ve instal.lat per defecte). És a dir, si fem ***sudo dnf install php-fpm*** (a Fedora) o ***sudo apt install php-fpm*** (a Ubuntu). Un cop instal.lat, la idea és configurar el servei Apache per a què sàpiga reenviar (a través del mòdul "proxy_fcgi") les sol.licituds de pàgines PHP al servei PHP-FPM. Els passos per aconseguir-ho són:

1) Activar (si no ho està ja) el mòdul "proxy_fcgi". A Ubuntu simplement es faria així: ***sudo a2enmod proxy_fcgi*** (s'activarà automàticament també el mòdul "proxy", dependència del primer). A Fedora aquest pas no cal perquè aquest mòdul ja s'activa per defecte gràcies a l'existència de l'arxiu "**/etc/httpd/conf.modules.d/00-proxy.conf**", proporcionat per la instal·lació per defecte de l'Apache.

2) Dir a l'Apache que redireccioni les peticions de pàgines PHP al "servidor PHP-FPM". Això es pot fer a nivell de servidor global o de VirtualHost. Si optem per la primera opció, a Ubuntu tenim el tros de configuració ja preparat (concretament dins d'un arxiu proporcionat pel paquet "php-fpm" i anomenat "**/etc/apache2/conf-available/phpX.Y-fpm.conf**"), però caldrà executar la comanda ***sudo a2enconf phpX.Y-fpm*** per activar-lo. A Fedora, en canvi, no cal fer res perquè el tros de configuració responsable ja es troba preparat (en aquest cas dins de l'arxiu "**/etc/httpd/conf.d/php.conf**", també proporcionat pel paquet "php-fpm") i llest per funcionar

NOTA: En realitat la línia important de la configuració "prefabricada" anterior és la que comença per "*SetHandler...*" Aquesta directiva indica el programa responsable al qual s'enviaran les peticions que concordin amb la secció *<FilesMatch>* on està inclosa (és a dir, el servidor "php-fpm"). Tal com es pot veure, la secció *<FilesMatch>* indica una expressió regular que representa tots els fitxers sol.licitats que tinguin una extensió com ".php" o similar. D'altra banda, en lloc de la directiva *SetHandler* ... també es podria haver fet servir la directiva *ProxyPassMatch* (amb el mateix valor de *SetHandler* com a únic paràmetre); per a més informació, es pot consultar http://wiki.apache.org/httpd/PHP-FPM

NOTA: Hi ha dos mètodes per comunicar un servidor web amb el PHP-FPM, tal com mostra la línia "SetHandler...": el mètode per defecte ("unix socket") assumeix que ambdós programes (servidor web i intèrpret PHP-FPM) estan executant-se a la mateixa màquina perquè és un mètode de traspàs de missatges local a través d'un "socket" compartit. Però si aquest mètode no funcionés per alguna raó, a la configuració "prefabricada" ja s'indica que, com a alternativa, es pot contactar via el protocol *fcgi://* amb un servidor PHP-FPM que igualment pot ser local (de fet, ja està així per defecte) però que també podria ser remot (perquè el protocol "fcgi" fa servir el mètode "tcp/ip socket", el qual ho permet). No obstant, si es fa servir aquest segon mètode, caldria configurar adientment aquest servidor PHP-FPM, cosa que no està feta d'entrada (en parlarem més endavant).

3) Iniciar els dos servidors: (***sudo systemctl start phpX.Y-fpm/php-fpm*** i ***sudo systemctl start apache2/httpd***). Cal assegurar- se també de què s'inicïin automàticament a cada reinici de la màquina (***sudo systemctl enable phpX.Y-fpm/php-fpm*** i ***sudo systemctl enable apache2/httpd***)

NOTA: A Fedora, en realitat només cal engegar explícitament el servei Apache perquè el servei PHP-FPM arrencarà automàticament sempre amb ell gràcies a l'existència d'un arxiu (proporcionat pel paquet "php-fpm") anomenat "**/usr/lib/systemd/system/httpd.service.d/php-fpm.conf**" amb el següent contingut:

```
[Unit]
Wants=php-fpm.service
```

Configuració del servidor PHP-FPM (*d'entrada no cal canviar-la*)

A Ubuntu

Dins de la carpeta "**/etc/php/*X.Y*/**" (on *X.Y* representa el nº de versió del paquet) es poden trobar unes quantes subcarpetes que contenen diferents fitxers de configuració de l'intèrpret PHP (els quals són fitxers de text pla en format INI amb un determinat contingut). En concret, les subcarpetes i fitxers més importants són:

*"**fpm/**" : Subcarpeta on hi ha un arxiu anomenat "**php.ini**" , arxiu principal de l'intèrpret PHP quan funciona en mode "FastCGI", i la subcarpeta "**conf.d/**", la qual inclou "trossos" de configuració en forma d' arxius "ini" (els noms dels quals comencen amb un número per definir l'ordre amb què es carregaran); aquests "trossos" sovint estan formats només per una línia de configuració, la línia *extension=nomModul*, la qual serveix per carregar un determinat mòdul PHP (ja que, a l'igual que passava amb el servidor Apache, l'intèrpret PHP també implementa una estructura de mòduls que es poden carregar per afegir una determinada funcionalitat al llenguatge que inicialment no té); així doncs, cada arxiu "ini" present dins de la carpeta "*conf.d/*" sol representar un mòdul determinat carregat (de fet, la seva presència dins de la carpeta "*conf.d/*" implica que efectivament estan carregats, perquè allà és on l'intèrpret PHP els va a buscar però en realitat aquests arxius "ini" són enllaços als arxius reals, que es troben ubicats en una altra subcarpeta, la subcarpeta "../**mods-available/**", la qual només serveix de magatzem -tant dels mòduls carregats com descarregats-; per tant, la forma "d'activar un mòdul PHP consistiria en crear un enllaç -amb *ln -s* - a l'arxiu "ini" desitjat dins de la carpeta "*conf.d/*"). Més endavant estudiarem alguns dels mòduls PHP que aporten les capacitats més interessants al llenguatge PHP

NOTA: Dins de l'arxiu de configuració "*php.ini*" podem establir moltes coses, per exemple: definir la zona horària que farà servir l'intèrpret a qualsevol script on hi apareguin dates (línia *date.timezone*); activar la possibilitat de veure al navegador els errors de sintaxi presents al nostre codi PHP (línia *display_errors*, molt recomanable posar-la a "On" quan estem desenvolupant la nostra web però molt important posar-la a "Off" en producció, per seguretat), etc. Altres línies interessants són: *error_reporting* (per definir el grau d'importància que ha de tenir l'error per tal de mostrar-lo o no), *error_log* (per definir la ruta del fitxer de registre on es guardaran els missatges d'error; només funciona si *log_error* val "On"), *max_input_time* (per definir el nº màxim de segons que un script es quedarà esperant dades provinents de l'exterior), *max_execution_time* (per definir el nº màxim de segons que un script romandrà executant-se), *memory_limit* (per definir el nº màxim de MB que un script pot consumir de RAM), *file_uploads* (si val "on" es permeten les pujades de fitxers al servidor web), *upload_max_filesize* (per definir la mida màxima de les pujades de fitxers), etc, etc. La llista d'opcions possibles dins del fitxer "php.ini" es troba a https://www.php.net/manual/en/ini.php

*"**cli/**" : Subcarpeta on hi ha un arxiu anomenat "**php.ini**", arxiu principal de l'intèrpret PHP quan aquest s'executa directament des d'un terminal (de fet, aquesta subcarpeta només existirà si s'ha instal·lat el paquet "**php-cli**", del qual en parlarem més endavant) i la subcarpeta "**conf.d/**", la qual inclou "trossos" de configuració en forma d'enllaços a arxius "ini" (els noms dels quals comencen amb un número per definir l'ordre amb què es carregaran); aquests "trossos" sovint estan formats només per una línia de configuració, la línia *extension=nomModul*, la qual serveix per carregar un determinat mòdul PHP (tal com hem explicat al paràgraf anterior). Igualment, els enllaços presents dins de la carpeta "*conf.d/*" apunten als arxius "ini" reals que es troben ubicats en la subcarpeta "../**mods-available/**"

En tot cas, en el cas d'utilitzar l'intèrpret PHP com a dimoni PHP-FPM, cal tenir present que qualsevol canvi que es guardi en algun dels fitxers i subcarpetes anteriors (o qualsevol càrrega/descàrrega que es faci d'algun mòdul PHP) no serà efectiu fins reiniciar el servidor PHP-FPM (o bé, de forma alternativa, executar la comanda ***sudo systemctl reload phpX.Y-fpm/sudo systemctl reload php-fpm***, la qual obliga al servidor a llegir de nou tots els arxius de configuració sense que calgui reiniciar).

A Fedora

Tots els intèrprets (de tipus "FastCGI", de terminal, etc) tenen el mateix arxiu de configuració general, (anomenat "**/etc/php.ini**" , amb un contingut similar a l'indicat a la "NOTA" del quadre anterior) i fan servir la mateixa subcarpeta de mòduls PHP (anomenada "**/etc/php.d**", on es troben els fitxers que contenen les línies *extension=nomModul* que ja hem vist que serveixen per carregar el/s mòdul/s PHP desitjats). I ja està (cal dir, que, en aquest cas, es pot veure que no existeix la possibilitat de carregar/descarregar mòduls PHP creant/eliminant enllaços, sinó que directament, per carregar un mòdul PHP caldrà escriure el fitxer amb el contingut adequat dins de *"/etc/php.d"* i per descarregar-lo s'haurà d'eliminar -o comentar- aquest contingut)

Més enllà de la configuració del servei PHP-FPM funcionant com a intèrpret PHP pròpiament dit (el que s'ha descrit al quadre anterior), aquest programa disposa d'un altre arxiu de configuració on s'estableixen les directives associades al seu comportament específicament donat com a servidor recolector de peticions de l'Apache i gestor de processos fills (com, de fet, diu el seu nom). Aquest "altre" arxiu a Ubuntu s'anomena "**/etc/php/*X.Y*/fpm/php-fpm.conf**" i a Fedora "**/etc/php-fpm.conf**". Es pot consultar un llistat de totes les directives possibles amb una breu explicació a https://www.php.net/manual/en/install.fpm.configuration.php; per exemple, per mencionar una directiva concreta, allà ens podem trobar, dins de la secció "*[global]*", la directiva *error_log* (que indica la ruta del fitxer de registre on es guardaran els eventuals missatges d'error del servidor PHP-FPM), entre d'altres. En tot cas, però, el més interessant d'aquest arxiu és la directiva *include* que agrega a aquesta configuració del servei PHP-FPM tots els arxius "*.conf" existents dins de la carpeta "**/etc/php/*X.Y*/fpm/pool.d**" (a Ubuntu) o a "**/etc/php-fpm.d/**" (a Fedora). En aquests arxius és on es configuren con funcionaran els processos fills que realment processen la petició provinent de l'Apache. Aquí es on podem especificar, per exemple, l'usuari i grup amb el què treballaran (via directiva *user* i *group*) i la manera com rebran les peticions (via directiva *listen*, ja sigui a través d'"unix sockets" -per defecte- o bé "tcp/ip sockets" -amb un valor tal com "127.0.0.1:9000", per exemple, etc)

NOTA: Per defecte, tant a Ubuntu com a Fedora existeix un únic arxiu de configuració per tots els processos fills, anomenat "**www.conf**". Però en servidors de hosting és molt freqüent tenir-ne varis, un per cada "virtual host" allotjat, de forma que els processos fills associat a un "virtual host" no afectin al de l'altre. Per aconseguir aquesta configuració, el que cal fer primer és configurar l'Apache per a què envïi les peticions al servidor PHP-FPM no de forma global sinò a cada "virtual host" per separat. Això s'aconsegueix incloent unes línies similars a les següents a cadascun dels arxius de configuració de cada "virtual host":

```
<FilesMatch "\.php$">
 SetHandler "proxy:unix:/run/php/nomVirtualHost.sock|fcgi://nomVirtualHost/"
</FilesMatch>
```

A partir d'aquí, cal copiar (com si fos una plantilla) l'arxiu "*www.conf*" a la mateixa carpeta i canviar el nom dels arxius-còpia per "*nomVirtualHost.conf*" així com la primera línia sense comentar d'aquests arxius de "*[www]*" a "[*nomVirtualHost*]". A més, seria recomanable canviar el valor de les línies *user* i *group*. Finalment, i això és obligatori, caldria modificar el valor de la directiva *listen* per a què apuntés al mateix arxiu ".sock" (o a la mateixa URL fcgi) que s'hagi indicat a la directiva *SetHandler* anterior.

NOTA: The following directives are ones that you will want to change based on site traffic and server resources (this is relevant only if *pm* directive is set to *dynamic*):

pm.start_servers – The number of PHP-FPM children that should be spawned automatically
pm.max_children – The maximum number of children allowed (connection limit)
pm.min_spare_servers – The minimum number of spare idle PHP-FPM servers to have available
pm.max_spare_servers – The maximum number of spare idle PHP-FPM servers to have available
pm.max_requests – Maximum number of requests each child should handle before re-spawning
pm.request_terminate_timeout – Max amount of time to process a request (similar to max_execution_time in php.ini)

These directives should all be adjusted based on the needs of each site. For instance, a really busy site with many, many, simultaneous PHP connections may need 3 servers each with 100 children, whilst a low-traffic site may only only need 1 server with 10 children. The thing to remember is that those children are only active whilst actually processing PHP, which could be a very short amount of time (possibly measured in milliseconds). Contrapositively, setting the numbers too high for the server to handle (in terms of available memory and processor) will result in poor performance when sites are under load. As with anything, tuning the pool requirements will take time and analysis to get it right. Més informació a https://medium.com/@sbuckpesch/apache2-and-php-fpm-performance-optimization-step-by-step-guide-1bfecf161534

Mòduls PHP (o com fer possible que l'intèrpret PHP pugui treballar amb SGBDs, entre altres coses)

Ja hem comentat anteriorment que la instal.lació per defecte de l'intèrpret PHP no proporciona tota la funcionalitat possible del llenguatge. Això fa que per poder utilitzar diferents aspectes del llenguatge PHP calgui instal.lar prèviament algun paquet extra, a mode d'"ampliació" de l'intèrpret bàsic. Això es fa per no engreixar d'entrada un intèrpret del qual potser no aprofitaríem moltes de les seves capacitats (capacitats que estan sempre igualment consumint recursos).

Aquests paquets extra, sigui la funcionalitat que sigui la que proporcionin, estan sempre compostos per dos elements principals: el mòdul en sí (que no és res més que una llibreria dinàmica; és a dir, en Windows, un arxiu "*.dll" i en Linux un arxiu "*,so" -el qual, en el cas concret d'Ubuntu, estarà ubicat a la carpeta "**/usr/lib/php**" i a Fedora estarà ubicat a la carpeta "**/usr/lib64/php/modules**"-) i l'arxiu "ini" corresponent a cada mòdul en qüestió (ubicats tots, com ja sabem, sota la carpeta "*/etc/php/X.Y/fpm/conf.d/*" -a Ubuntu- o "*/etc/php.d*" -a Fedora-) i que serveixen, gràcies a la directiva *extension=nomModul*, per carregar el mòdul associat efectivament en RAM.

Una de les funcionalitats importants que no vénen en la instal.lació per defecte de l'intèrpret PHP és la de poder contactar amb servidors de bases de dades. Per tant, si volem manipular bases de dades allotjades en aquest tipus de servidors, hauríem d'instal.lar obligatòriament un paquet extra. Per exemple, per treballar amb bases de dades MySQL hem d'instal.lar el paquet "**php-mysql**" (a Ubuntu) o "**php-mysqlnd**" (a Fedora); per treballar amb bases de dades SQLite hem d'instal.lar el paquet "**php-sqlite3**" (només a Ubuntu però a Fedora es pot usar el paquet genèric per múltiples bases de dades anomenat "**php-pdo**", que d'altra banda, ja ve instal·lat per defecte a Ubuntu) ; per treballar amb bases de dades PostgreSQL hem d'instal.lar el paquet "**php-pgsql**" (a Ubuntu i Fedora), etc, etc. En qualsevol cas, després de la instal·lació del paquet pertinent haurà d'aparèixer al sistema un nou fitxer "ini" que, gràcies a la seva presència, farà que es carregui automàticament el mòdul recentment descarregat al proper (i propers) reinicis del servei PHP-FPM (o, si l'intèrpret PHP està funcionant com a mòdul de l'Apache, al/s proper/s reinici/s del servei Apache) i, per tant, es pugui a partir de llavors fer ús de la nova funcionalitat afegida.

Un símptoma clar de no haver fet el pas anterior és, per exemple, voler escriure un codi PHP contenint diferents funcions d'accés i manipulació de bases de dades i obtenir, en voler-lo mostrar en un navegador, un error del tipus "*Undefined function xxxx*". Tal com diu aquest missatge, l'error indica que l'intèrpret PHP no reconeix una funció específica, la qual només serà entesa amb la instal.lació de l'"ampliació" corresponent.

Instal.lació de l'intèrpret PHP a Windows

Més enllà de la versió de l'intèrpret PHP "pelat" per Windows (https://windows.php.net/download), en aquest sistema operatiu disposem de paquets integrals que instal.len tots els components necessaris (servidor web -normalment Apache-, intèrpret PHP amb multitud d'extensions activades, servidor de base de dades -normalment MySQL-, etc) per començar a desenvolupar una web PHP completa amb múltiples funcionalitats disponibles ja per defecte. Exemples són:

WampServer (http://www.wampserver.com)
MAMP (https://www.mamp.info)
XAMPP (https://www.apachefriends.org)
EasyPHP (https://www.easyphp.org)
Laragon (https://laragon.org)
DevilBox (http://devilbox.org) -Basat en contenidors Docker-

Primeres proves

Sigui quina sigui la manera que haguem escollit per instal.lar l'intèrpret PHP, el primer que farem serà comprovar que funcioni. Per això haurem de crear dins del DocumentRoot d'algun VirtualHost actiu del nostre servidor Apache (per defecte això és la carpeta "/var/www/html") un arxiu de texte anomenat "index.php" (cal vigilar que aquest nom estigui indicat a la directiva *DirectoryIndex*, tot i que la instal·lació de l'intèrpret PHP ja estableix per defecte aquest valor a l'arxiu "php.conf" de la configuració de l'Apache) amb el següent contingut:

```
<html><body><?php  phpinfo();  ?></body></html>
```

Com es pot comprovar, la manera d'incrustar codi PHP dins del codi HTML és mitjançant les etiquetes especials "*<?php*" (inici codi PHP) i "*?>*" (final codi PHP). El que fa la funció *phpinfo()* -no oblideu el punt i coma del final...en aquest sentit, la majoria de sintaxi de PHP és heretada de C- és generar una taula informativa de color lila amb multitud de dades de configuració de l'intèrpret PHP. Ara no ens aturarem en esbrinar el significat d'aquestes dades (encara que, per exemple, es pot veure fàcilment a l'apartat "Server/API" quin mode d'intèrpret PHP està funcionant): només ens interessa veure si aquesta taula apareix o no. Per fer això, haurem d'escriure a la barra del navegador una direcció tal com http://nostra.ip (o http://nostra.ip/index.php, si voleu). Si es veu la taula lila, tot ha anat perfecte.

NOTA: ¿Què es veuria al navegador si substituïm la funció *phpinfo();* del codi anterior per la línia *print Date("Y/m/d");*?

Ús de l'intèrpret PHP via terminal

Tot i que no és el més habitual, els scripts PHP poden ser executats invocant l'intèrpret directament des d'un terminal (com si fossin scripts Bash o Python) mitjançant la comanda *php* (la qual està disponible instal.lant prèviament el paquet "**php-cli**"). Concretament, un cop instal.lat aquest intèrpret PHP de consola, podem obrir un terminal i provar les següents comandes per comprovar que funcioni tot correctament:

php -i : Mostra la configuració actual de l'intèrpret (similar a com ho fa la funció *phpinfo();*), que en aquest cas és l'intèrpret de consola. Recordeu que, en aquest cas, aquesta configuració a Ubuntu està emmagatzemada dins de l'arxiu general "*/etc/php/X.Y/cli/php.ini*" (i, eventualment, dins d'arxius més específics, corresponents a la configuració de mòduls concrets, ubicats a la carpeta "*/etc/php/X.Y/cli/conf.d*") i a Fedora està dins de l'arxiu general *"/etc/php.ini"* (i, eventualment, dins d'arxius específics ubicats a la carpeta "*/etc/php.d*").

NOTA: Una altra comanda que ens informa sobre les característiques de l'intèrpret PHP actual és *php -m* , la qual mostra la llista de mòduls PHP ("llibreries") actualment carregades (és a dir, disponibles per utilitzar als nostres scripts). Recordem que que estiguin carregades (o no) depèn de si la directiva "*extension=nomModul*" està inclosa (o no) dins de la configuració de l'intèrpret PHP (arxiu "php.ini" i arxius dins de la carpeta "conf.d")

NOTA: Una altra comanda que ens informa sobre les característiques de l'intèrpret PHP actual és *php --ini* , la qual mostra la llista de les rutes dels fitxers de configuració actualment llegits (el general més els específics de mòduls concrets). D'altra banda, si es volgués executar l'intèrpret PHP amb un fitxer de configuració diferent indicat per nosaltres "ad-hoc", això es pot fer amb el paràmetre -c, així: *php -c /ruta/carpeta/php.ini*

NOTA: A vegades no es vol canviar de fitxer de configuració general però sí que es vol canviar el valor de certa opció per una determinada execució de l'intèrpret. En aquest cas es pot indicar el nom i valor desitjat d'aquesta opció amb el paràmetre *-d*. Per exemple, l'opció *display_errors*, si val "on", mostra tots els valors que l'intèrpret s'ha trobat per pantalla (normalment aquesta opció està posada a "off" per motius de seguretat); així doncs, si volguéssim activar-la per l'execució actual caldria fer: *php -d display_errors=on script.php*

NOTA: En aquest sentit, un paràmetre també interessant perquè serveix per cercar possibles errors de sintaxi en un script és *-l : php -l script.php*

php -r "codi php" : Executa el codi PHP directament escrit entre les cometes. Per exemple, la comanda *php -r "phpinfo();"* executa la funció pròpia de PHP anomenada *"phpinfo();"*. Es poden indicar vàries ordres una darrera l'altra; per exemple, *php -r "phpinfo(); echo('Hola');"* executa primer "*phpinfo();*" i després la funció "*echo*" (que simplement serveix per mostrar per pantalla el missatge indicat).

php script.php : Executa el codi PHP que estigui escrit dins del fitxer indicat (en aquest cas, "script.php"). Els fitxers que continguin codi PHP solen tenir ".php" com extensió (encara que això no és obligatori). El que sí que han de complir és el següent requisit per a què l'intèrpret els pugui executar correctament: la primera línia del seu contingut ha de ser *<?php* i la darrera ha de ser *?>*. Per exemple, un possible contingut de l'arxiu "script.php" podria ser:

```
<?php
phpinfo();
echo("Hola");
?>
```

NOTA: Opcionalment es pot indicar el paràmetre *-f* abans del nom de l'script a l'hora d'executar la comanda. No té efectes pràctics, però.

NOTA: Si l'script necessités fer servir paràmetres, el/s seu/s valor/s concrets es poden escriure directament darrera del nom de l'script a l'hora d'executar la comanda

NOTA: Si la primera línia de l'script (just abans de *<?php*) és aquesta: *#!/usr/bin/php* (on la ruta és la del binari PHP al teu sistema, ruta que es pot conèixer executant *which php*) en sistemes Linux es podria convertir l'script en executable (així: *chmod +x script.php*). D'aquesta manera, a l'hora d'executar l'script només caldria indicar la seva ruta i prou com qualsevol altre script Bash o Perl (així: *./script.php*)

php -a : Posa en marxa una consola interactiva dins de la qual es poden escriure les ordres PHP que es desitgin i interpretar-se en temps real. Admet autocompletatge de comandes mitjançant la pulsació de la tecla tabulador i navegació per les comandes ja executades mitjançant la pulsació de les tecles de cursos endavant-endarrera (aquest historial es guarda al fitxer "**~/.php_history**"). Per sortir-ne cal executar la comanda *quit* (o *exit*)

NOTA: Hi ha un parell d'opcions de configuració especialment interessants per l'entorn interactiu.Una és *cli.prompt=unvalor*, la qual canvia el "prompt" de l'entorn; l'altra és *cli.pager=less* , la qual indica el programa paginador extern que es farà servir quan la sortida mostrada sigui més llarga que la longitud de la pantalla. Aquestes opcions es poden indicar a l'arxiu "*php.ini*" pertinent o bé directament a l'entorn (on afectaria només a la sessió actual); en aquest darrer cas, però, cal precedir-les del símbol "#" (així: *#cli.pager=less*) per a què l'intèrpret les entengui com opcions i no com a codi.

php -v : mostra la versió de l'intèrpret PHP
php -h : mostra tota la llista de paràmetres possibles de l'intèrpret PHP de consola

Per saber més com utilitzar l'intèrpret en aquest mode, es pot consultar https://www.php.net/manual/en/features.commandline.php , on s'explica, entre altres coses, com recollir els valors dels paràmetres indicats al terminal dins del codi PHP (via variables especials *$argc* i *$argv*), o com esperar-se a rebre dades del teclat de forma interactiva (via STDIN), etc.

EXERCICIS:

1.- Instal·la i configura tot el necessari per a què al teu sistema (Ubuntu o Fedora) tinguis un intèrpret PHP-FPM funcionant i un servidor Apache que en faci ús. Comprova-ho fent el necessari per veure via navegador la taula generada per la funció *phpinfo();*

El fet que l'intèrpret PHP funcioni en mode "FastCGI" vol dir que les peticions d'execució de codi PHP que rep des de l'exterior (normalment provinents d'un servidor web) es transmeten fent servir internament aquest protocol específic. Tot i que no és gaire habitual, es poden enviar aquestes peticions manualment, per estudiar millor aquest protocol. És el que farem al següent exercici, substituïnt l'Apache per un altre "client FastCGI" (concretament, una comanda de terminal que executarem puntualment per enviar una petició d'execució concreta)

2.-a) A la mateixa màquina on s'està executant el servidor PHP-FPM, instal·la el paquet "libfcgi0ldbl" (a Ubuntu) o el paquet "fcgi" (a Fedora) per tal de disposar del client FastCGI *cgi-fcgi* A partir d'aquí (i suposant que el valor de la directiva *listen* present a l'arxiu "/etc/php/X.Y/fpm/pool.d/www.conf" -a Ubuntu- o "/etc/php-fpm.d/www.conf" -a Fedora- no hagi canviat de l'establert per defecte -és a dir, que el servidor PHP-FPM estigui escoltant en un "socket" local, concretament a "*/run/php/phpX.Y-fpm.sock*" en Ubuntu i a "*/run/php-fpm/www.sock*" en Fedora-), prova la següent comanda (la qual s'ha d'executar com l'usuari propi de l'Apache -"www-data" a Ubuntu i "apache" a Fedora- perquè és l'únic -a banda de "root"- que té permisos, gràcies a una ACL pertinent, de llegir i escriure sobre el "socket" utilitzat -si la connexió fos TCP/IP això no caldria-, i a l'entorn de la qual s'han definit prèviament un conjunt de variables que estableixen quin codi PHP concret s'hi vol enviar i mitjançant quin mètode)...:

*A Ubuntu: *sudo -u www-data SCRIPT_NAME=/var/www/codi.php SCRIPT_FILENAME=/var/www/codi.php REQUEST_METHOD=GET cgi-fcgi -bind -connect /run/php/phpX.Y-fpm.sock*
*A Fedora: *sudo -u apache SCRIPT_NAME=/var/www/codi.php SCRIPT_FILENAME=/var/www/codi.php REQUEST_METHOD=GET cgi-fcgi -bind -connect /run/php-fpm/www.sock*

...on "var/www/codi.php" pot ser un codi PHP qualsevol com per exemple: *<?php phpinfo(); ?>* ¿Què obtens com a resultat?

> NOTA: El paràmetre *-connect* de la comanda *cgi-fcgi* serveix per indicar el punt de connexió on estarà escoltant el servidor PHP-FPM. En el cas d'haver-lo configurat per escoltar a través de la xarxa (assignant a la línia *listen* abans comentada un valor com *ip.serv.php.fpm:nºport*), en aquest paràmetre caldrà indicar llavors el mateix valor, "*ip.serv.php.fpm:nºport"* El paràmetre *-bind* indica que es connectarà a un servidor ja en marxa (és obligatori indicar-lo)
>
> NOTA: És obligatori indicar la ruta absoluta de l'script a enviar. En teoria hi ha una variable d'entorn reconeguda per *fcgi-cgi* anomenada *DOCUMENT_ROOT=* que serveix per indicar la ruta base a partir de la qual indicar la ubicació dels scripts (per exemple, així DOCUMENT_ROOT=/var/www SCRIPT_NAME=/codi.php SCRIPT_FILENAME=/codi.php) però no funciona (veure https://github.com/hollodotme/fast-cgi-client/wiki/Background-Info-FastCgiClient-Version-2.6.0 per més informació). Alternativament a l'ús d'aquesta variable d'entorn, també existeix la directiva *chroot* dins del fitxer "www.conf", la qual té el mateix significat que ella.
>
> NOTA: Si s'estableix la directiva *pm.status_path* del fitxer "www.conf" al valor "/status" (tot i que podria ser qualsevol altre), el servidor PHP-FPM reconeixerà automàticament el valor de "/status" (o el que s'hagi indicat) assignat a les variables *SCRIPT_NAME=* i *SCRIPT_FILENAME=* com una senyal de què ha de retornar un conjunt d'estadístiques de funcionament intern del servidor (temps que porta encès, nombre de peticions rebudes i ateses amb èxit, etc...per més infomració podeu consultar https://easyengine.io/tutorials/php/fpm-status-page).
>
> NOTA: La petició "/status" anterior admet una variable d'entorn especial anomenada *QUERY_STRING=* que pot valer "full" (per obtenir més informació detallada), "*json*" (per obtenir-la en format JSON) o "*full&json*" (per les dues coses alhora), "xml", "html", etc. De fet, aquesta variable *QUERY_STRING=* també es pot usar quan es demana executar un codi PHP personalitzat per tal de transmetre'l variables
>
> NOTA: Si s'estableix la directiva *ping.path* del fitxer "www.conf" al valor "/ping" (tot i que podria ser qualsevol altre), el servidor PHP-FPM reconeixerà automàticament el valor de "/ping" (o el que s'hagi indicat) assignat a les variables *SCRIPT_NAME=* i *SCRIPT_FILENAME=* com una senyal de què ha de retornar una resposta "pong".
>
> NOTA: Per saber més sobre el protocol "FastCGI", consulteu https://fastcgi-archives.github.io

3.-Atura el servei PHP-FPM (per tal de confirmar que, efectivament, no l'estiguis fent servir) i instal·la el necessari per a què al teu sistema (Ubuntu o Fedora) tinguis un intèrpret PHP de consola. A continuació, obre un terminal i executa les següents comandes. ¿Què veus a la sortida de cadascuna d'elles?

a) *php -i | less*

b) *php -r "phpinfo(); echo('Hola');"*

c) *php ~/codi.php* , on "codi.php" és un fitxer a la nostra carpeta personal amb el següent contingut:

```
<?php
phpinfo();
echo("Hola");
?>
```

d) *php -a* (i, al shell intern que apareix, executar les següents comandes, una per una: *phpinfo(); echo("Hola");* i *exit* -o *quit*-)

e) *php -v*