



UFCD 0788

Instalação e administração de servidores WEB

Síntese

Planear, instalar e gerir um servidor intranet e Internet em ambiente web.

António Sousa
243758375@formacao.iefp.pt

Conteúdo

Serviços cliente/servidor.....	5
Objetivos	5
Considerações iniciais	5
Tipos de serviços	5
Resumo	7
Arquitetura cliente/servidor	8
Objetivos	8
Considerações iniciais	8
Arquitetura.....	8
Resumo	10
Ambientes de servidor	11
Objetivos	11
Considerações iniciais	11
Soluções	11
Resumo	15
Servidores <i>web</i>	16
Objetivos	16
Servidor <i>web</i>	16
Implementação	16
Configuração	19
Servidor de conteúdo dinâmico (PHP).....	23
Servidor de bases de dados (MySQL).....	25
Configuração do servidor <i>web</i> em ambiente Windows.....	27
Resumo	38
Linguagem PHP	39
Sintaxe e marcadores de comandos	39
Comentários.....	39
Maiúsculas e minúsculas.....	40
Tipos de dados	40
Tipo Inteiro.....	40
Tipo de Vírgula Flutuante.....	41
Tipo Booleano	42

Tipo String	43
Strings com plicas.....	43
Strings com aspas.....	43
Função GetType().....	44
Converter tipos de dados.....	45
Testar o tipo de variável.....	46
Declarar e usar variáveis	47
Contexto de variáveis.....	47
Operadores	49
Operadores de atribuição	50
Operadores aritméticos	50
Ordem de Precedência dos Operadores Aritméticos	51
Operadores Lógicos, Bitwise e Relacionais.....	52
Operadores de deslocamento de bits.....	52
Operadores relacionais	53
Operador lógicos.....	53
Ordem de precedência dos operadores relacionais	54
Operador de Concatenação de Strings	54
Estruturas de controlo condicional.....	54
Controlo Condicional Simples	54
Controlo condicional composto.....	55
Controlo condicional encadeado	55
Estrutura de controlo switch	57
Estruturas de repetição.....	57
Estrutura de repetição while	57
Estrutura de repetição do ... while.....	58
Estrutura de repetição for.....	59
Estrutura de repetição foreach.....	59
Instrução break	60
Instrução continue	61
Declarar e manipular arrays.....	62
Criar um array	62
Manipular os elementos de um array.....	63
Remover um array ou os seus elementos.....	63
Processar os dados recebidos de um formulário HTML	64
Array \$_POST e \$_GET	66

Ligação a uma base de dados MySQL	70
Efetuar a ligação à base de dados	71
Função die()	74
Inserir dados na base de dados MySQL	74
Efetuar uma consulta SQL na base de dados e devolver dados à página HTML	77
Servidores FTP	80
Objetivos	80
Considerações iniciais	80
Instalação	80
Configuração	80
Resumo	84
Servidores DNS	85
Objetivos	85
Considerações iniciais	85
Instalação	86
Configuração	86
Cache de DNS	87
Resumo	89
Servidores de correio eletrónico	90
Objetivos	90
Considerações iniciais	90
Instalação	90
Configuração	92
Resumo	93
Servidor de partilha de arquivos e servidor de impressão	94
Objetivos	94
Considerações iniciais	94
Instalação	95
Configuração	95
Ferramentas de administração de servidores e serviços	101
Objetivos	101
Considerações iniciais	101
Instalação	102
Configuração	103
Resumo	104
Referências	106

Serviços cliente/servidor

Objetivos

Compreender os conceitos sobre os serviços que as aplicações cliente/servidor oferecem.

Considerações iniciais

Os sistemas cliente/servidor têm como objetivo oferecer, através de um computador robusto, uma série de serviços que ficam acessíveis aos clientes das mais diversas formas possíveis.

O funcionamento de um sistema cliente/servidor consiste numa situação onde um cliente envia um pedido para um servidor, que a processa, e envia a resposta para o cliente. Com base nessa resposta, o cliente pode enviar novas requisições.

O servidor, como se mencionou anteriormente, normalmente consiste num computador com grande poder de processamento, grande poder de armazenamento e capacidade de transmitir informações em alta velocidade, através da rede. É comum o mesmo servidor oferecer vários serviços, visto que o computador tem essas condições. Assim, é eliminada a necessidade de aquisição de vários servidores. Dessa forma, o serviço é executado no servidor, numa porta predeterminada e sempre que um cliente realizar uma requisição naquela porta, o servidor responde de forma adequada. Noutras palavras, o servidor, ao iniciar o serviço, fica a aguardar pedidos de utilizadores.

O cliente é o responsável por dar início à troca de informações, pois é ele que envia um pedido que será processado pelo servidor. Esse pedido pode ser enviado das mais diversas formas possíveis. É muito comum que exista uma aplicação específica para cada caso, conforme será mostrado posteriormente.

Também é muito comum, que sistemas cliente/servidor funcionem de forma que a comunicação entre o cliente e o servidor seja realizada através de rede, pois em muitos casos o cliente e o servidor não estão no mesmo computador. A Figura 1.1 mostra um exemplo básico de um serviço cliente/servidor a funcionar através da internet.

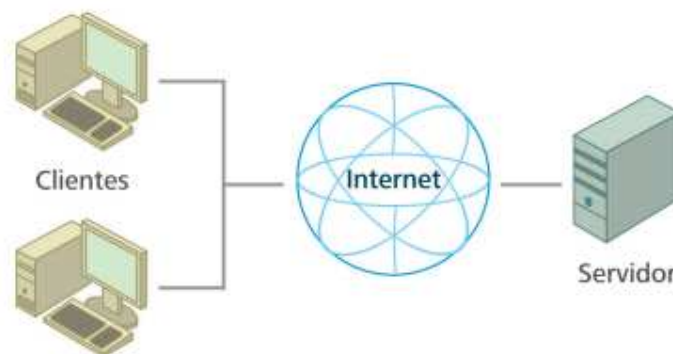


Figura 1-Comunicação cliente/servidor

Tipos de serviços

O sistema cliente/servidor mais utilizado provavelmente é o serviço *web* que consiste no acesso de páginas de internet nos navegadores.

O servidor web possui um serviço a ser executado, no qual a porta 80, ou 8080 (nas configurações padrão) fica a ser monitorizada, e quando alguma informação é enviada para o servidor através dessa porta, ele interpreta-a conforme a configuração do serviço. Nesse caso, o cliente, que é um utilizador qualquer a aceder a uma página web através do seu computador, envia um pedido, que é o endereço

da página que ele quer carregar. O servidor, por sua vez, processa essa informação, devolvendo a página web para o cliente. O cliente pode então interagir com essa página através de formulários, enviando um novo pedido para o servidor, iniciando o processo todo novamente. Neste caso é fácil perceber que o cliente e o servidor não estão no mesmo computador. Isso significa que a comunicação entre eles é realizada através de rede. Mas, num pedido como esse, o próprio servidor pode fazer o papel de cliente, acedendo a outro serviço que está a ser executado dentro do mesmo servidor. Por exemplo, quando o utilizador utiliza um serviço de autenticação, ele envia o pedido de uma página de internet que será acessível apenas se as informações que ele enviou para o servidor estiverem presentes numa base de dados no servidor. Nesse caso, o cliente envia o pedido para o servidor que precisa consultar as informações do utilizador na base de dados. Então um pedido é enviado do servidor web para o servidor de base de dados. O servidor de base de dados devolve para o servidor web as informações solicitadas. O servidor web utiliza as informações fornecidas pelo servidor de base de dados para montar a página web solicitada pelo utilizador e a envia para o mesmo.

É possível perceber que na comunicação entre o servidor web e o servidor de base de dados, existe uma grande possibilidade de os dois serviços estarem a ser executados no mesmo computador e, nesse caso, a comunicação deles é apenas entre os serviços dentro do servidor, não dependendo da rede.

É visível que nesse ambiente de cliente/servidor, o cliente normalmente utiliza uma aplicação com a qual pode interagir das mais diversas formas para aceder os serviços do servidor. No caso de páginas web, o utilizador utiliza o navegador. Outro serviço cliente/servidor mencionado anteriormente é o servidor de bases de dados. No exemplo citado o servidor de bases de dados é acedido pelo servidor web, sem ser através de uma interface de utilizador, ou seja, é um serviço que comunica com outro serviço. E, no caso, o utilizador não comunica diretamente com o servidor de bases de dados.

Pode existir, entretanto, uma aplicação qualquer que aceda diretamente a base de dados. Como exemplo, é possível utilizar uma aplicação de automação comercial. Nesse caso, existe um servidor de bases de dados, no qual estão armazenadas todas as informações da empresa como registo dos clientes, funcionários, produtos, vendas, etc. A aplicação, por sua vez, foi desenvolvida especificamente para aquele fim. Quando um utilizador a utiliza, está a comunicar diretamente com o servidor de bases de dados através da rede. A Figura 2 mostra a estrutura de uma aplicação cliente/servidor que acede diretamente a uma base de dados.



Figura 2-Cliente a receber informações de um servidor de bases de dados

Outro serviço cliente/servidor muitas vezes utilizado está presente em aplicativos de IM (Instant Messenger), que nada mais são que as aplicações de mensagens instantâneas, como WhatsApp, Skype, Facebook Messenger, entre outros. Nesse caso, existe também uma aplicação específica que é utilizado pelo utilizador. Essa aplicação, por sua vez, liga-se ao servidor e oferece para o utilizador uma série de funcionalidades.

O servidor deve ser capaz de processar os pedidos enviados pelos utilizadores e, para isso, ele utiliza várias regras presentes nos protocolos de comunicação. Nos casos de mensagens instantâneas via texto, o protocolo utilizado para a comunicação é o TCP (*Transmission Control Protocol*). Esse protocolo define que numa comunicação entre dois dispositivos, a comunicação deve ser aberta, existe a troca de informações e, posteriormente, essa comunicação é fechada.

É possível fazer uma analogia com a comunicação através de um rádio amador. Uma comunicação através de rádio amador inicia-se quando um utilizador chama verbalmente outro, mas a comunicação entre eles só acontece quando o utilizador que foi chamado, responder. Quando o utilizador chamado responder, eles iniciam a comunicação entre si, sendo que um utilizador fala e quando quer indicar que terminou de falar, ou seja, terminou de enviar informações, utiliza a palavra “câmbio”. O outro utilizador, por sua vez, ao ouvir a palavra “câmbio” entende que todas as informações foram recebidas e começa a falar, a enviar suas informações. É comum também, o uso da palavra “entendido”, que tem como objetivo confirmar se o outro utilizador conseguiu compreender as informações enviadas. Para terminar a comunicação, utiliza-se a frase “câmbio e desligo”. Uma comunicação através de protocolo TCP funciona de forma semelhante. Um dispositivo inicia a comunicação enviando um pacote que tem como objetivo estabelecer uma comunicação entre os dois dispositivos. O outro dispositivo responde, e a troca de informações inicia. Cada vez que um dispositivo envia uma informação, o dispositivo que a recebeu, responde com uma confirmação de que a informação foi recebida, semelhante ao “entendido”. Quando a troca de informações termina, os utilizadores trocam um pacote que indica o término da comunicação, semelhante ao “câmbio e desligo”.

Esse processo é realizado para garantir que as informações não foram perdidas durante a comunicação e, caso um dispositivo enviar uma informação e não receber a confirmação de que ela foi recebida pelo destinatário, ele envia-a novamente.

Num serviço de troca de mensagens de texto, esse processo é utilizado. Quando um utilizador envia uma mensagem, o destinatário retorna um pacote informando que essa mensagem foi recebida, garantindo a integridade das informações.

Vários aplicativos de mensagens instantâneas oferecem o serviço de chamadas de vídeo que consistem numa comunicação de videoconferência. Esse tipo de comunicação utiliza o protocolo UDP (*User Datagram Protocol*) que, diferentemente do protocolo TCP, não solicita a confirmação da recepção das informações enviadas. Isso significa que um utilizador envia informações sem parar, independentemente de o destinatário recebê-las ou não. Isso acontece porque nesse tipo de comunicação, se houver a verificação das informações, a comunicação torna-se incompreensível. O mesmo acontece em chamadas de voz pela internet, o VOIP.

É muito comum, na elaboração de programas, a utilização de *sockets* para realizar a comunicação entre dois computadores. O *socket* é uma estrutura capaz de fazer com que um computador execute um serviço numa determinada porta, operando como servidor, da mesma forma, proporciona que um cliente se comunique com essa porta. Assim é possível desenvolver um programa de computador do tipo cliente/servidor para as mais diversas necessidades.

Resumo

Foram apresentados os serviços cliente/servidor, os seus tipos, e serviços utilizados atualmente. Para explicar o funcionamento de uma comunicação realizada através de um protocolo TCP, foi feita uma analogia à comunicação através de rádio amador.

Arquitetura cliente/servidor

Objetivos

Compreender o funcionamento da arquitetura de sistemas cliente/servidor.

Considerações iniciais

Conforme a visto anteriormente, os sistemas cliente/servidor normalmente são compostos por um servidor, no qual um ou mais serviços estão a ser executados. O servidor geralmente é um computador com grande capacidade de processamento, armazenamento e de transmissão de dados pela rede em alta velocidade. O outro componente de um sistema cliente/servidor é o cliente que vai aceder o serviço, ou serviços, disponibilizado(s) pelo servidor. Claro que o acesso pode ser realizado por mais de um cliente simultaneamente.

O serviço oferecido pelo servidor opera numa porta definida pela aplicação que o manipula e pode ser acedida das mais diversas formas. Por exemplo, uma aplicação cliente/servidor muito comum é o serviço web, no qual o servidor possui páginas web que são acedidas pelo cliente através do navegador. Existem casos onde o servidor faz também o papel de cliente, quando o mesmo redireciona uma consulta que recebeu de um cliente para outro servidor. Um exemplo disso é o servidor DNS. Quando o utilizador escreve o endereço de um *website* no seu navegador, o servidor DNS pega nesse endereço e o converte num endereço IP para descobrir qual computador na *Internet* essa *website* está armazenado. Porém, se esse servidor DNS não possuir o endereço de IP do site requisitado, ele vai consultar outro servidor DNS, que possui mais informações que ele. Nessa situação, o servidor que recebeu o primeiro pedido do *site*, faz o papel de cliente, quando envia a solicitação para outro servidor DNS.

O cliente de uma aplicação cliente/servidor tem, na grande maioria das vezes, uma *interface* personalizada pela qual ele acede os serviços oferecidos pelo servidor. Essa *interface* personalizada pode ser desenvolvida especificamente para um propósito, como uma *interface* que acede a uma base de dados específica no servidor. Cada base de dados possui uma estrutura diferente e mesmo que a base de dados esteja no mesmo servidor, será necessária uma *interface* ligeiramente diferente para aceder a base de dados. Por causa desse contexto, é possível oferecer uma série de serviços personalizados, visto que a *interface* é desenvolvida para um fim específico. Existem casos ainda de *interfaces* genéricas. Estas têm como objetivo aceder um serviço disponível por um servidor, entretanto, como são genéricas, devem ser compatíveis com a estrutura do serviço oferecido pelo servidor. No exemplo do servidor web, a *interface* utilizada pelo utilizador é o navegador. Atualmente, existem vários tipos de navegadores que têm o mesmo objetivo: aceder o serviço oferecido pelo servidor web. Os navegadores existentes atualmente são ligeiramente parecidos, pois eles têm de ser compatíveis com uma grande quantidade de protocolos utilizados pelas páginas de *Internet*. Porém, eles diferem entre si porque uns oferecem algumas funcionalidades a mais, como extensões e plugins que têm como objetivo melhorar a sua usabilidade.

Arquitetura

A arquitetura cliente/servidor é dividida em quatro níveis diferentes, de acordo com sua complexidade e generalidade. A Figura 3 mostra a arquitetura cliente/servidor simples. Nesse modelo, existe a comunicação apenas entre um cliente e um servidor. O servidor fica a aguardar que o cliente inicie a comunicação.



Figura 3-Arquitetura cliente/servidor simples

Já a Figura 4 mostra a arquitetura cliente/servidor em dois níveis, centrada no servidor. Nesse modelo, vários clientes acedem o serviço, ou serviços, apenas num servidor.



Figura 4-Arquitetura cliente/servidor em dois níveis - centrada no servidor

A Figura 5 traz o modelo cliente/servidor em dois níveis, centrado no cliente. Nesse caso, pelo facto de o cliente não poder perceber a presença de outros clientes no sistema e ter acesso a vários serviços no servidor, ele pode ter a impressão de que existem vários servidores.



Figura 5-Arquitetura cliente/servidor em dois níveis - centrada no cliente

Na realidade, o que acontece é que vários clientes comunicam com vários servidores e vários servidores comunicam com vários clientes simultaneamente, conforme ilustra a Figura 6.



Figura 6-Arquitetura cliente/servidor - comunicação entre vários clientes e vários servidores

É importante salientar que os modelos apresentados nas Figuras 4, 5 e 6 representam o mesmo modelo, visto sob pontos de vista distintos.

No modelo par a par, o cliente também faz o papel de servidor, e o servidor faz o papel de cliente. Na arquitetura simples, é possível perceber que o servidor fica a executar o serviço, porém ele não inicia nenhum processo, fica a aguardar o cliente enviar um pedido. Quem inicia a comunicação entre o cliente e o servidor é o cliente. Já na arquitetura par a par, pelo fato de, em certos momentos um cliente se comportar como servidor, e um servidor se comportar como cliente, qualquer um pode iniciar uma comunicação. Nesse modelo, qualquer dispositivo da rede pode tornar-se um cliente ou um servidor. A Figura 7 representa a arquitetura par a par.

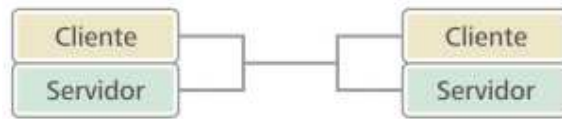


Figura 7-Arquitetura cliente/servidor par a par

O último modelo apresentado é o de comunicação em vários níveis. Nele, existe a situação onde o servidor algumas vezes faz o papel de cliente, enviando um pedido para outro servidor. A Figura 8 representa esse modelo.

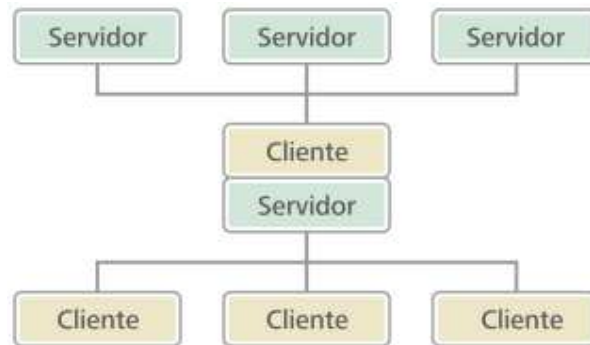


Figura 8-Arquitetura cliente/servidor multinível

Deve-se lembrar que cada modelo é adequado para um contexto, não existe um melhor ou um pior. Cada modelo se encaixa num caso. Um estudo deve ser realizado para averiguar o modelo que vai atender às necessidades de implementação do sistema.

Resumo

Começou-se por apresentar considerações iniciais sobre sistemas cliente/servidor, trazendo o seu conceito e exemplos da sua aplicação. Apresentou também, as diferentes arquiteturas de sistemas cliente/servidor juntamente com o seu conceito e funcionamento. Vale ressaltar que cada arquitetura cliente/servidor adequa-se a um caso diferente. Por esse motivo, a instalação de um sistema cliente/servidor deve ser planejada para não desperdiçar recursos.

Ambientes de servidor

Objetivos

Conhecer os ambientes existentes em servidores.

Considerações iniciais

Como abordado anteriormente, os servidores são capazes de automatizar e melhorar a estrutura de um sistema ou de uma rede de computadores. O objetivo de um servidor é oferecer um ou mais serviços. O servidor, na grande maioria das vezes, é um computador com grande capacidade de processamento e armazenamento. Destinado apenas à execução de um ou mais serviços. Dessa forma, esse computador é utilizado somente por um utilizador, quando este está a realizar alguma manutenção ou instalação de um novo serviço. Nesse caso, esse utilizador é o administrador do sistema e não utilizador de um recurso da rede. Noutras palavras, o servidor nunca é utilizado como se fosse uma máquina para uso de um recurso da rede. Por exemplo, um servidor *web* nunca é utilizado para aceder a páginas de internet. É costume evitar o uso desnecessário do servidor, pois assim economizam-se recursos e diminui a possibilidade de acontecerem problemas no servidor.

Os servidores têm grande participação numa rede de computadores e podem desempenhar as mais diversas atividades. Numa rede SOHO (*Small Office/Home Office*), por exemplo, que muitas vezes é composta apenas por poucos computadores e com acesso à internet, possui um servidor no próprio *router*, que é responsável por fazer a distribuição do acesso à internet para os computadores da rede.

Entre os serviços encontrados com grande frequência numa rede de computadores, estão os que têm como objetivo otimizar os recursos da rede como armazenamento prévio de informações de páginas de internet, armazenamento de arquivos, servidores de e-mail, servidores de base de dados, serviços que têm como objetivo melhorar a segurança da rede como um todo, serviços para implementar políticas de segurança na rede.

É possível afirmar que os servidores estão presentes na grande maioria das redes e que são de grande importância. Eles oferecem formas de melhorar a administração da rede e meios de implementar serviços em larga escala.

Soluções

Entre as mais diversas soluções implementadas em servidores, algumas são mais utilizadas por oferecerem recursos comuns a vários casos, conforme a lista a seguir.

- **Servidores *web*** - têm como objetivo armazenar uma ou mais páginas de internet. É através desse servidor que um *website* será disponibilizado para os utilizadores, e na grande maioria das vezes, ele também armazena um servidor de base de dados que armazenará as informações utilizadas por formulários do *website*. A Figura 9 mostra o acesso a um servidor *web*.

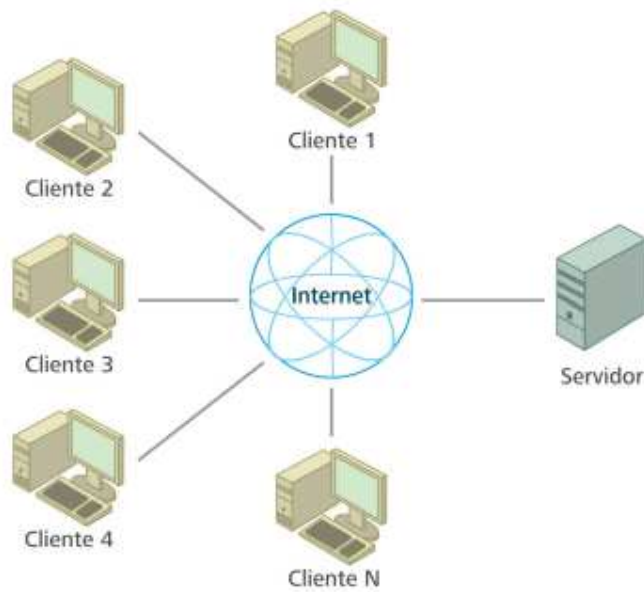


Figura 9-Acesso a um servidor web

- **Servidor de arquivos** - esse servidor armazena qualquer arquivo numa pasta compartilhada na rede. Esses arquivos podem ser acessíveis a todos os utilizadores da rede, ou seja, é uma pasta pública. Também podem ser acessíveis apenas utilizadores registados a partir de utilizador e palavra-passe. Nesse segundo caso, visto que o acesso é realizado por utilizador e palavra-passe, pode-se fazer uma restrição de acesso a apenas um grupo de utilizadores. Também é possível disponibilizar uma pasta em modo leitura. Isso significa que os utilizadores poderão aceder à pasta e ler o conteúdo dela, mas não poderão inserir arquivos nessa pasta nem alterar os arquivos que nela estão. A Figura 10 mostra o acesso a um servidor de arquivos.

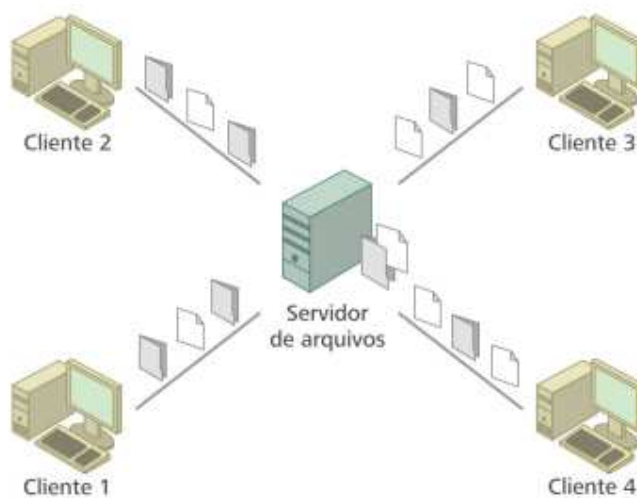


Figura 10-Acesso a um servidor de arquivos

- **Servidor de impressão** - como o próprio nome diz, esse servidor está ligado fisicamente a uma impressora e permite que os utilizadores da rede utilizem a impressora remotamente. É possível, através da configuração do servidor, definir cotas de impressão e limitar o uso do recurso para um grupo de utilizadores. A Figura 11 mostra um ambiente com um servidor de impressão.

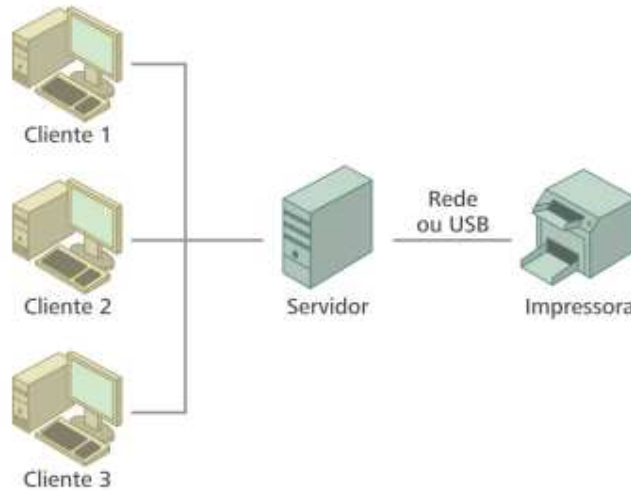


Figura 11-Ambiente com um servidor de impressão

- **Servidor de correio eletrónico** - muitas empresas têm o costume de implementar os seus próprios servidores de correio eletrónico (e-mail) para evitar que os seus funcionários utilizem os seus e-mails pessoais para o trabalho. A Figura 12 mostra um ambiente com servidor de correio eletrónico *Postfix*.

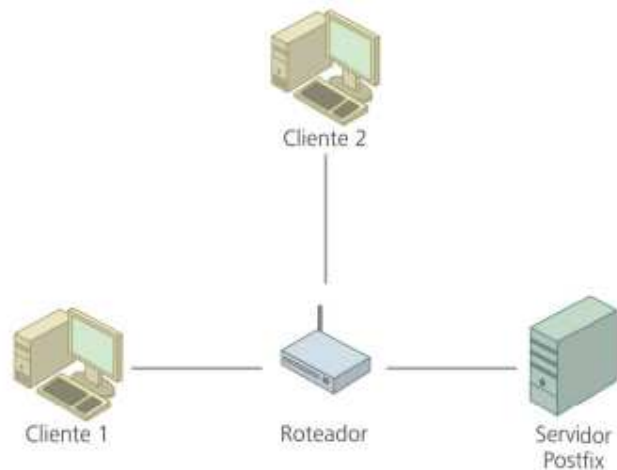


Figura 12-Ambiente com um servidor de correio eletrónico *Postfix*

- **Servidor DNS (*Domain Name System*)** - tem como objetivo traduzir endereços de websites. Quando o utilizador escreve o endereço de um *website* que ele quer aceder, esse pedido é enviado para o servidor DNS. Ele, por sua vez, consulta uma tabela para descobrir o endereço IP do computador que armazena o *website* solicitado. Caso ele encontre o endereço IP, ele envia o pedido da página web para o endereço encontrado. Caso ele não encontre, ele envia o mesmo pedido para um servidor que possua mais informações. É comum utilizar o servidor DNS em conjunto com uma aplicação para armazenamento de informações em *cache*. Isso funciona da seguinte forma: quando um utilizador acede a uma página de *Internet*, o servidor

armazena a estrutura dessa página, como o posicionamento dos elementos no ecrã. Quando esse *website* for acedido por qualquer outro utilizador da rede, essa informação é carregada do servidor de cache de DNS e não do servidor *web*. Assim, economiza-se a banda da ligação com a Internet. A Figura 13 mostra um pedido a um servidor DNS. No passo 1, o cliente envia o endereço *web* com o qual deseja se comunicar, no caso “*icann.org*”; ao servidor DNS, no passo 2 devolve o endereço IP onde o site solicitado está armazenado, 192.0.2.0, neste exemplo.

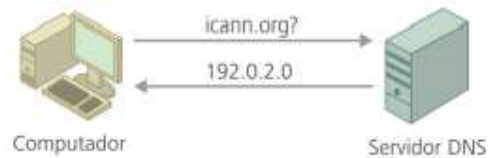


Figura 13-Funcionamento de um servidor DNS

- **Servidor de bases de dados** - várias aplicações utilizam dados de utilizadores e transações no decorrer do seu uso. Esses dados ficam armazenados em bases de dados que possuem uma estrutura avançada de segurança e administração, bem como ferramentas de integridade de dados e ferramentas que permitem que esses dados possam ser acedidos das mais diversas formas.
- **Servidor de *internet/firewall*** – em redes de médio e grande porte, é comum o uso de um servidor que irá compartilhar a ligação da *Internet* com os computadores da rede. Isso é feito, porque com ele é possível aplicar uma enorme quantidade de ferramentas de segurança e filtros de conteúdo indevido, da mesma forma que limita o uso de banda. É possível afirmar que o objetivo geral de uma *firewall* é permitir que qualquer pedido realizado dentro da rede seja enviado para a *Internet* (caso os filtros de conteúdo permitam) e que toda e qualquer pedido enviado da *Internet* para dentro da rede seja bloqueada. A Figura 14, apresentada em baixo como exemplo, mostra uma rede protegida por uma *firewall*.

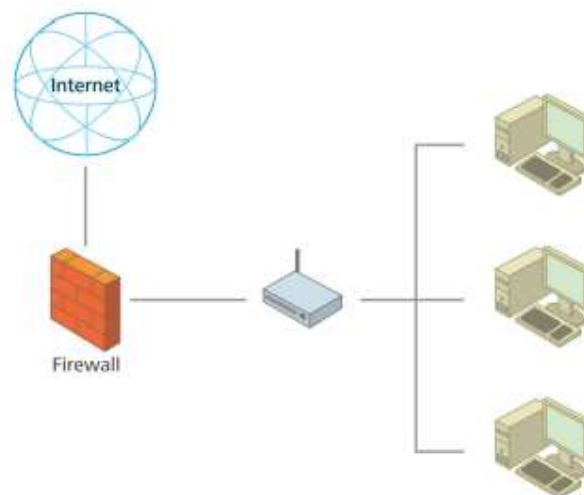


Figura 14-Rede protegida por firewall

Obviamente, os servidores possuem as mais diversas funcionalidades que incluem autenticação de utilizadores na rede, partilha de arquivos e recursos, entre outras. As vantagens de se utilizar um servidor é concentrar a administração do recurso. Dessa forma, caso seja necessário alterar uma

configuração de um serviço utilizado pela rede, basta alterar a configuração desse serviço no servidor, reiniciar o serviço e, automaticamente, todos os outros computadores da rede passarão a operar de acordo com a nova configuração do serviço.

Outra grande vantagem do uso de servidores é que em alguns casos, ele concentra também informações sobre os utilizadores da rede. Se ele não existisse, existiriam várias cópias da mesma informação em vários computadores da rede. Um exemplo simples disso é a autenticação dos utilizadores nos computadores. Imagine que existe uma rede de uma empresa que possui uma enorme quantidade de utilizadores. Cada utilizador possui a sua mesa com o seu computador e nesse computador ele possui vários arquivos que utiliza para trabalhar. Agora imagine que esse computador apresentou um problema e deve ser substituído. Todos os dados do utilizador e seus arquivos serão perdidos. Caso exista uma cópia dessas informações, elas devem ser colocadas no novo computador. Esse problema não parece ser grande, quando visto como um problema de um utilizador, mas imagine se esse problema ocorrer simultaneamente com uma grande quantidade de utilizadores.

Para resolver esse problema, pode-se utilizar um servidor que possua as informações de todos os utilizadores da rede, bem como uma pasta compartilhada com cada um deles. Assim, quando um utilizador se autentica num computador qualquer da rede, esse computador irá verificar no servidor se o utilizador e palavra-passe inseridos estão corretos. Caso estejam, irá carregar todas as configurações que o utilizador utiliza no seu computador, da mesma forma que irá carregar o acesso à sua pasta que está, na verdade, armazenada no servidor. Dessa forma, qualquer utilizador pode utilizar qualquer computador da rede, que suas informações estarão disponíveis a qualquer momento.

Esse cenário também pode vir acompanhado com políticas de segurança implementadas na rede. Um grupo de utilizadores pode ter acesso limitado dentro do sistema de uma empresa. Por exemplo, o setor de recursos humanos pode ter acesso somente ao sistema que gere os recursos humanos, enquanto o presidente dessa empresa terá acesso a todo o sistema. Assim, caso um funcionário do setor de recursos humanos se autentique no computador utilizado pelo presidente, com seu utilizador e palavra-passe, as permissões que esse utilizador terá são as de um funcionário do setor de recursos humanos, e somente terá acesso à parte de recursos humanos do sistema. Já caso o presidente da empresa se autentique num computador do setor de recursos humanos, ele terá acesso total ao sistema da empresa. É visível, portanto, que as políticas de acesso e segurança estão vinculadas ao utilizador e não ao computador. Entretanto, vale lembrar que essas políticas também são aplicáveis aos computadores. Nesse caso, mesmo que o presidente da empresa se autentique num computador do setor de recursos humanos, ele terá acesso apenas ao que é permitido para um computador do setor de recursos humanos.

Resumo

Apresentaram-se ambientes de servidor, características de servidor de rede, sua utilização e sua utilidade para o sistema como um todo.

Também foram apresentados diversos tipos de servidores largamente utilizados, dada a importância dos serviços por eles prestados.

Servidores web

Objetivos

Conhecer o funcionamento de servidores, web e servidores DNS, instalá-los e configurá-los num sistema operativo GNU/Linux.

Servidor web

Servidores *web* (também conhecidos como servidores HTTP) são os responsáveis por processar pedidos HTTP (*Hypertext Transfer Protocol*) que, por sua vez, são as responsáveis pela troca de pedidos de páginas de Internet entre um utilizador e um servidor *web*.

Por outras palavras, o servidor *web* é quem disponibiliza as páginas *web* de cada site. Por exemplo, cada vez que um utilizador abre um navegador, escreve o endereço de um *website* e pressiona um botão para carregá-la, o navegador envia um pedido HTTP para o endereço escrito pelo utilizador. Esse *website*, disponibilizado por um servidor *web*, recebe o pedido HTTP e responde com a página *web* solicitada. Essa página é o conteúdo de um arquivo com extensão .html ou .htm que é interpretada pelo navegador e exibida para o utilizador.

Normalmente, um servidor *web* não é implementado sozinho, pois se for, ele será capaz de disponibilizar para os utilizadores apenas páginas HTML (*Hypertext Markup Language*), que permitem somente a exibição estática de conteúdo. Isso significa que cada informação presente na website deve ser inserida manualmente, página por página, alterando-lhe o código HTML. Dessa forma, não é possível a construção de páginas com recursos como formulários, registos e gestão de conteúdos. É comum os servidores *web* virem acompanhados de servidores PHP (*Hypertext PreProcessor*) ou servidores ASP (*Active Server Pages*), que permitem o uso de conteúdo dinâmico, juntamente com um servidor de base de dados (como o MySQL) responsável pelo armazenamento de informações, ou seja, a persistência de dados.

É muito comum, num servidor *web*, encontrar-se o que é conhecido como LAMP, que significa Linux+Apache+MySQL+PHP que são, respetivamente, o sistema operativo presente no servidor, o servidor *web*, o servidor de base de dados e o servidor de conteúdo dinâmico, capaz de interpretar páginas dinâmicas desenvolvidas na linguagem PHP. Vale lembrar que todas as aplicações citadas são gratuitas. Isso significa que qualquer utilizador pode baixá-los e utilizá-los sem nenhum custo. Ainda assim, as aplicações Apache, MySQL e PHP podem ser instalados num sistema operativo Windows.

Já as páginas dinâmicas desenvolvidas na linguagem ASP são normalmente encontradas em servidores Microsoft Windows, pois a linguagem também é desenvolvida pela Microsoft e por esse motivo, num computador com uma versão para servidores do sistema operativo da Microsoft, o conjunto necessário para ter um servidor *web* completo (servidor *web*, servidor de bases de dados e servidor de conteúdo dinâmico) já vem pré-instalado, bastando apenas configurá-lo. Dessa forma, as aplicações para as finalidades de servidor *web* e persistência são também desenvolvidos pela Microsoft. É possível instalar um servidor de conteúdo dinâmico num servidor Linux, porém a solução não é desenvolvida pela Microsoft.

Implementação

Geralmente, instalar um servidor *web* num sistema operacional Linux, consiste na instalação dos pacotes necessários para a sua execução, e na inicialização de um serviço que fará com que o servidor seja efetivamente ativado. No caso de um servidor *web* (e também na grande maioria das aplicações semelhantes no Linux), quando a aplicação é instalada, ele já traz consigo versões dos arquivos de configuração que contém uma configuração padrão que permite que o serviço seja executado com as

configurações básicas. Portanto, basta alterar o(s) arquivo(s) de configuração para personalizar o serviço de acordo com a necessidade de cada caso.

Basicamente, existem duas formas de instalar os pacotes necessários para um servidor *web* completo num sistema operativo Linux. A maneira mais fácil é usar um gestor de pacotes. Nesse caso, utiliza-se um gestor de pacotes presente na distribuição Linux (o gestor de pacotes pode variar de distribuição para distribuição). As vantagens de sua utilização são:

- A instalação das bibliotecas necessárias é realizada automaticamente.
- As versões dos pacotes são sempre as mais estáveis.
- Praticidade, pois basta uma linha de comando.
- Menor probabilidade de erros, pois os pacotes são designados para aquela distribuição.

A única desvantagem marcante desse método é que geralmente, em repositórios de pacotes designados para servidores, a versão dos pacotes é sempre a mais estável, o que significa que ela provavelmente não é a mais atual. Isso é feito para evitar conflito entre bibliotecas e possíveis instabilidades que pacotes novos possam apresentar. Dessa forma, caso seja necessária alguma funcionalidade existente apenas numa versão que não está presente nos repositórios, esse pacote precisa ser instalado manualmente e de forma cautelosa.

A outra maneira de instalar pacotes, mais complicada, consiste em baixá-los com os códigos fontes das aplicações e instalá-los manualmente, compilando um por um, inclusive as bibliotecas. A única vantagem deste método, como se descreveu anteriormente, é que os pacotes presentes nos repositórios de servidores são os mais estáveis e assim, uma nova versão que apresente funcionalidade necessária precisa ser instalada manualmente.

Em contrapartida, as desvantagens desse método são:

- Apresenta grande probabilidade de problemas de compilação e incompatibilidade de bibliotecas.
- É trabalhoso, pois cada pacote deve ser compilado e instalado manualmente.
- Requer mais conhecimento do profissional que irá instalar os pacotes.
- Pode haver problemas nas instalações desse tipo em um servidor que já esteja operante.

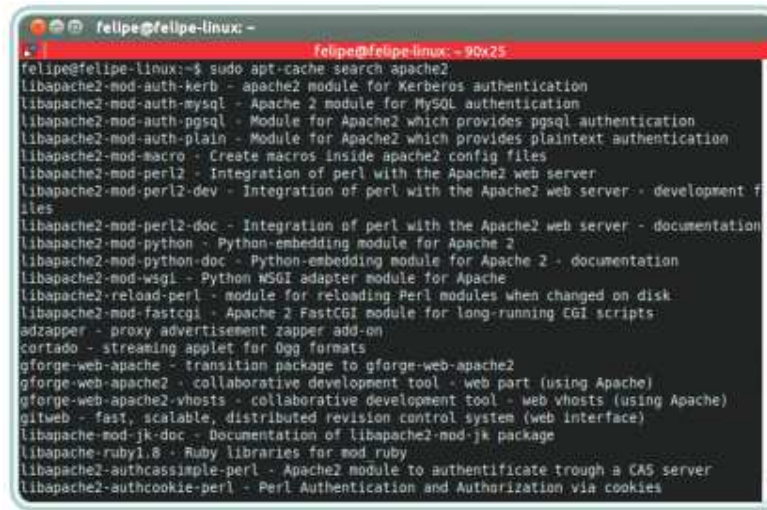
Pelo fato de o objetivo não ser instalação de pacotes, o método utilizado será através do gestor de pacotes. Para isso, a distribuição Linux escolhida como base será o Ubuntu, que é baseado no Debian e que utiliza o gestor de pacotes apt-get. Essa distribuição foi escolhida, porque é uma das mais utilizadas no mundo atualmente, possuindo uma versão para computadores pessoais com uma interface gráfica amigável, inclusive para o gestor de pacotes e uma versão para servidores, que não possuem interface gráfica, eliminando assim a possibilidade de problemas causados por mau uso do sistema operativo.

Para instalar um pacote, utilizando o comando apt-get, basta seguir a seguinte estrutura: **apt-get install <nome-do-pacote>**.

No entanto, para descobrir o nome do pacote, podemos utilizar o sistema de buscas do apt-get, através do apt-cache (Versão 10.04 ou mais recente do Ubuntu). Ex.: **apt-cache search <nome-do-pacote>**. A Figura 15 mostra um exemplo da execução do comando de pesquisa. Encontrado o termo *apache2*, significa que todos os pacotes que tiverem o termo “*apache2*” no nome ou descrição e que estiverem presentes no repositório aparecerão como resultado. A partir disso, é possível filtrar os pacotes necessários para a instalação de qualquer coisa em um sistema Linux e, conforme já

mentionado, ao instalar uma aplicação através do gestor de pacotes, todos os outros pacotes que forem requisitados do que está a ser instalado, serão instalados também.

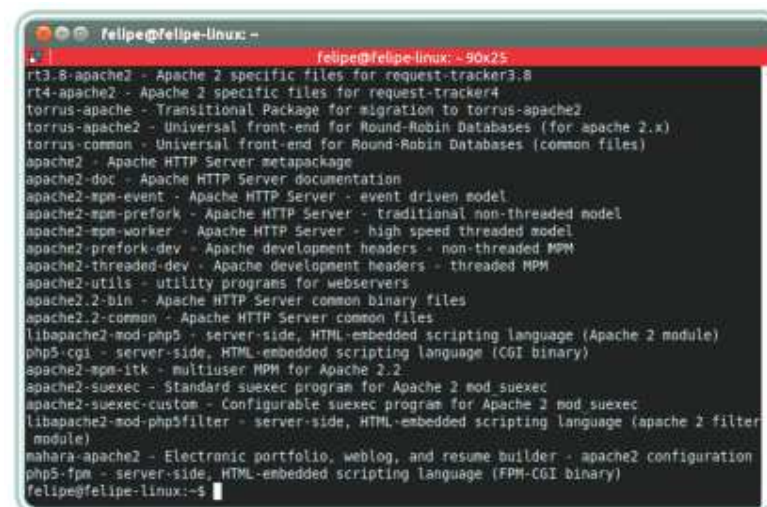
Uma das vantagens da utilização de servidores com sistema operativo Microsoft Windows, é que eles têm assistentes gráficos que facilitam a instalação, configuração e gestão de pacotes e serviços, porém esse tipo de sistema operativo normalmente tem um custo elevado e, por utilizarem uma interface gráfica, consomem mais recursos do computador. Outro problema que pode ocorrer é a má utilização do sistema operativo, pois ele oferece uma interface gráfica semelhante à versão para computadores pessoais. Já os sistemas operativos Linux, quando instalados em servidores, são mais leves (nesse caso geralmente não possuem interface gráfica) e gratuitos, porém mais difíceis de configurar.



```
felipe@felipe-linux: -
felipe@felipe-linux: ~$ sudo apt-cache search apache2
libapache2-mod-auth-kerb - apache2 module for Kerberos authentication
libapache2-mod-auth-mysql - Apache 2 module for MySQL authentication
libapache2-mod-auth-pgsql - Module for Apache2 which provides pgsql authentication
libapache2-mod-auth-plain - Module for Apache2 which provides plaintext authentication
libapache2-mod-macro - Create macros inside apache2 config files
libapache2-mod-perl2 - Integration of perl with the Apache2 web server
libapache2-mod-perl2-dev - Integration of perl with the Apache2 web server - development files
libapache2-mod-perl2-doc - Integration of perl with the Apache2 web server - documentation
libapache2-mod-python - Python-embedding module for Apache 2
libapache2-mod-python-doc - Python-embedding module for Apache 2 - documentation
libapache2-mod-wsgi - Python WSGI adapter module for Apache
libapache2-reload-perl - module for reloading Perl modules when changed on disk
libapache2-mod-fastcgi - Apache 2 FastCGI module for long-running CGI scripts
adzappper - proxy advertisement zipper add-on
kortado - streaming applet for Ogg formats
gforge-web-apache - transition package to gforge-web-apache2
gforge-web-apache2 - collaborative development tool - web part (using Apache)
gforge-web-apache2-vhosts - collaborative development tool - web vhosts (using Apache)
gitweb - fast, scalable, distributed revision control system (web interface)
libapache-mod-jk-doc - Documentation of libapache2-mod-jk package
libapache-ruby1.8 - Ruby libraries for mod ruby
libapache2-authcasimple-perl - Apache2 module to authenticate through a CAS server
libapache2-authcookie-perl - Perl Authentication and Authorization via cookies
```

Figura 15-Exemplo do comando apt-cache

Entre os pacotes mostrados no resultado da pesquisa, é possível identificar os pacotes do servidor HTTP Apache2, como mostra a figura 16.

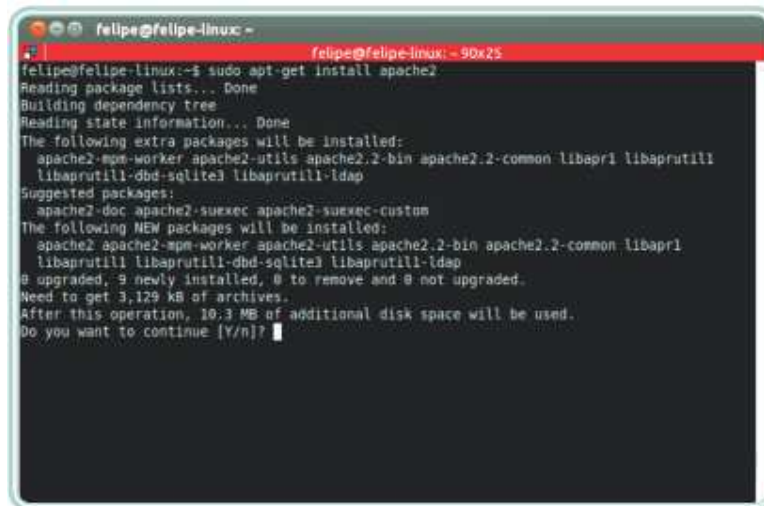


```
felipe@felipe-linux: -
felipe@felipe-linux: ~$ apt-cache search apache2
rt3.8-apache2 - Apache 2 specific files for request-tracker3.8
rt4-apache2 - Apache 2 specific files for request-tracker4
torrus-apache - Transitional Package for migration to torrus-apache2
torrus-apache2 - Universal front-end for Round-Robin Databases (for apache 2.x)
torrus-common - Universal front-end for Round-Robin Databases (common files)
apache2 - Apache HTTP Server metapackage
apache2-doc - Apache HTTP Server documentation
apache2-mpm-event - Apache HTTP Server - event driven model
apache2-mpm-prefork - Apache HTTP Server - traditional non-threaded model
apache2-mpm-worker - Apache HTTP Server - high speed threaded model
apache2-prefork-dev - Apache development headers - non-threaded MPM
apache2-threaded-dev - Apache development headers - threaded MPM
apache2-utils - utility programs for web servers
apache2.2-bin - Apache HTTP Server common binary files
apache2.2-common - Apache HTTP Server common files
libapache2-mod-php5 - server-side, HTML-embedded scripting language (Apache 2 module)
php5-cgi - server-side, HTML-embedded scripting language (CGI binary)
apache2-mpm-itk - multiuser MPM for Apache 2.2
apache2-suexec - Standard suexec program for Apache 2 mod suexec
apache2-suexec-custom - Configurable suexec program for Apache 2 mod suexec
libapache2-mod-php5filter - server-side, HTML-embedded scripting language (apache 2 filter module)
mahara-apache2 - Electronic portfolio, weblog, and resume builder - apache2 configuration
php5-fpm - server-side, HTML-embedded scripting language (FPM-CGI binary)
felipe@felipe-linux: ~$
```

Figura 16-Pacotes do Apache2 no resultado da pesquisa

Após a execução do comando `apt-cache`, é possível verificar os pacotes presentes no repositório e escolher qual(is) será(ão) instalado(s). Para instalar um pacote, ou mais de um, é utilizado o comando `apt-get`, conforme a seguinte estrutura: `apt-get install <nome-do-pacote>`

Nesse caso, é obrigatório que seja informado exatamente o nome do pacote. É possível instalar mais de um pacote de cada vez, inserindo os nomes dos pacotes um após o outro, separados apenas por espaços. A Figura 17 mostra a execução do comando `apt-get`.

A terminal window titled 'felipe@felipe-linux: ~' with a red title bar. The terminal shows the command 'felipe@felipe-linux:~\$ sudo apt-get install apache2' and its output. The output includes 'Reading package lists... Done', 'Building dependency tree', 'Reading state information... Done', and a list of extra packages to be installed: 'apache2-mpm-worker', 'apache2-utils', 'apache2.2-bin', 'apache2.2-common', 'libapr1', and 'libaprutil1'. It also lists suggested packages: 'apache2-doc', 'apache2-suexec', and 'apache2-suexec-custom'. The terminal then shows the packages to be installed, the disk space requirements, and asks for confirmation to continue. The user has entered 'Y' for yes.

```
felipe@felipe-linux:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 9 newly installed, 0 to remove and 0 not upgraded.
Need to get 3,129 kB of archives.
After this operation, 10.3 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

Figura 17-Execução do comando `apt-get` com o pacote encontrado na pesquisa

Configuração

Como se mencionou anteriormente, as ferramentas desenvolvidas para servidores Linux, na sua instalação, trazem consigo uma configuração padrão, capaz de fazer com que o serviço seja executado normalmente. Entretanto, por ser uma configuração padrão, pode não atender às necessidades do local onde ela está a ser instalada e, por isso, alterações são necessárias para fazer com que a aplicação funcione conforme o desejado.

Em sistemas Linux, a configuração dessas ferramentas é realizada através da alteração de um arquivo de configuração que é lido pela aplicação e interpretado, conforme uma linguagem não específica pré-determinada. Esse arquivo, na maioria das vezes, vai conter a configuração padrão, bastando alterá-lo em determinados lugares para personalizá-lo. Assim, não é necessário o conhecimento prévio da linguagem utilizada no arquivo.

Portanto, basta editar o arquivo com um editor de textos qualquer. Caso o sistema possua interface gráfica, pode-se utilizar um editor gráfico. Por exemplo, na interface de utilizador Gnome, o editor de textos padrão é o Gedit (Figura 18) e na interface de utilizador KDE, é o Kate (Figura 19) ou basta executar uma aplicação que permita acesso ao terminal de comandos e usar um editor de texto que utilize apenas recursos do terminal de comandos (os editores VI, Vim (Figura 20) e Nano (Figura 21) são os mais conhecidos e utilizados). Caso o sistema não possua interface gráfica, apenas os editores desenvolvidos para o terminal de comandos poderão ser utilizados. Neste documento, o editor utilizado será o Nano, por possuir uma interface de fácil manuseamento.

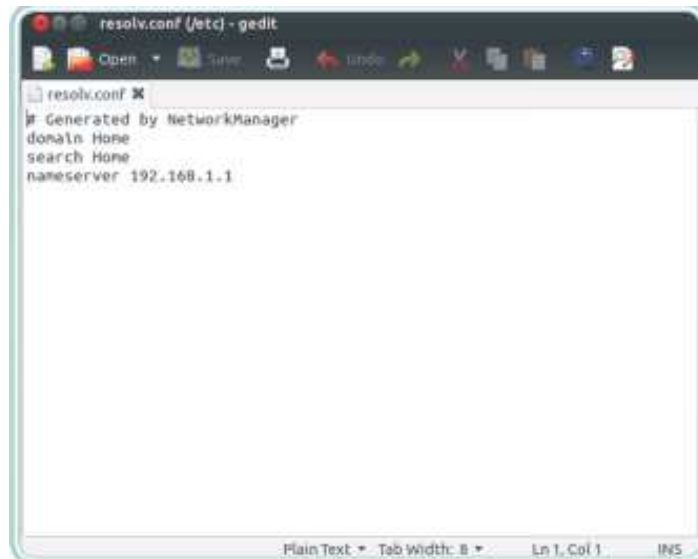


Figura 18-Editor de texto Gedit

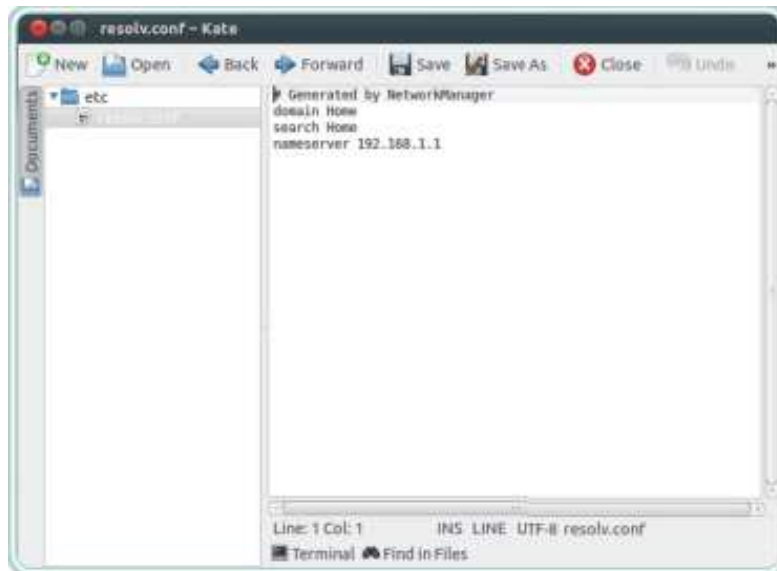


Figura 19-Editor de texto Kate

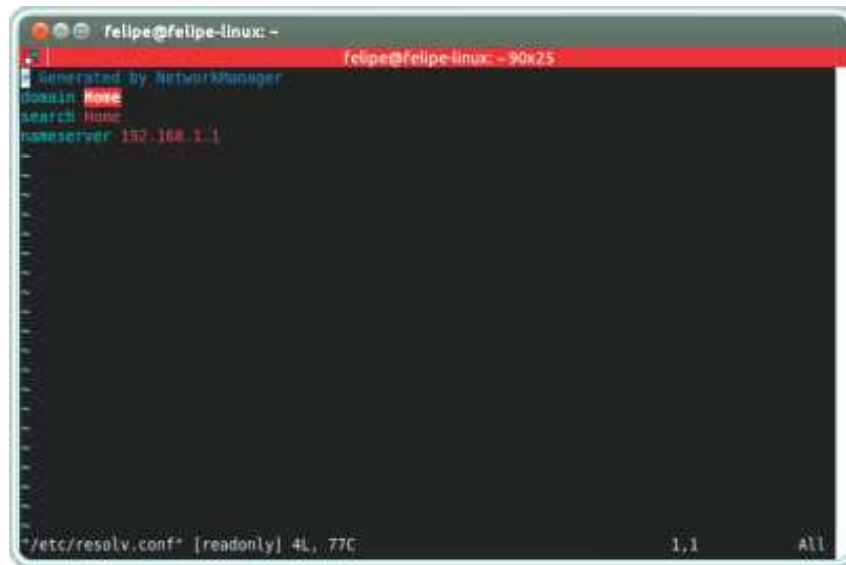
A screenshot of a terminal window showing the Vim text editor. The window title is 'felipe@felipe-linux: -'. The editor is displaying the file '/etc/resolv.conf'. The content of the file is: 'Generated by NetworkManager', 'domain: home', 'search: home', and 'nameserver 192.168.1.1'. The status bar at the bottom shows '/etc/resolv.conf' [readonly] 4L, 77C, and the cursor is at line 1, column 1.

Figura 20-Editor de texto Vim


A screenshot of a terminal window showing the Nano text editor. The window title is 'felipe@felipe-linux: -'. The editor is displaying the file '/etc/resolv.conf'. The content of the file is: 'Generated by NetworkManager', 'domain: home', 'search: home', and 'nameserver 192.168.1.1'. The status bar at the bottom shows 'GNU nano 2.2.6' and 'File: /etc/resolv.conf'. A warning message 'Read 4 lines (Warning: No write permission)' is displayed. The bottom status bar contains various keyboard shortcuts: 'G Get Help', 'O WriteOut', 'R Read File', 'Y Prev Page', 'K Cut Text', 'C Cur Pos', 'X Exit', 'J Justify', 'W Where Is', 'V Next Page', 'U UnCut Text', and 'T To Spell'.

Figura 21-Editor de texto Nano

Para editar um arquivo no Linux, utilizando o editor Nano, basta executar o comando “nano” seguido do nome do arquivo, juntamente com o endereço completo no qual ele se encontra. Por exemplo, para editar o arquivo resolv.conf, que fica na pasta /etc do Linux, que possui o endereço /etc/resolv.conf, basta executar o seguinte comando: **nano /etc/resolv.conf**

É importante lembrar que os arquivos que estão dentro da pasta /etc são arquivos de configuração de aplicativos e, portanto, um utilizador sem permissão do administrador será capaz de abrir e editar o arquivo, mas não será capaz de sobrescrever o seu conteúdo. Para isso, num sistema Ubuntu Linux, basta executar o comando “sudo” antes do comando “nano”, como se mostra a seguir: **sudo nano /etc/resolv.conf**

Assim, o sistema pedirá que o utilizador insira a sua senha e permitirá que ele sobrescreva o arquivo.

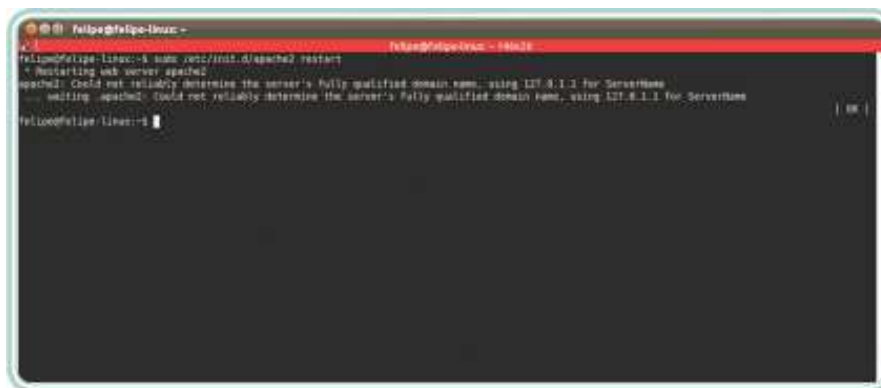
O comando `sudo` pode não aparentar um aumento de segurança, mas o seu objetivo é que o utilizador esteja ciente do que está a fazer. Por exemplo, caso um utilizador esteja a alterar um arquivo através do comando `sudo`, ele está ciente de que aquele arquivo é importante para o funcionamento do sistema e de aplicações e por isso são necessárias permissões de administrador para editá-lo. Assim o utilizador poderá tomar precauções para evitar danos ao sistema.

Uma vez instalado o servidor HTTP, basta iniciar o serviço que ele já estará a funcionar, porém caso seja necessário editar seu arquivo de configurações, ele está presente no diretório `/etc/apache2/httpd.conf`. Um dos parâmetros que normalmente é alterado é a porta na qual o servidor irá funcionar. A porta padrão é a 80. Caso a porta seja alterada, deve-se informar no navegador ao aceder o servidor. Por exemplo, caso a porta for alterada para 8080, para aceder o servidor HTTP é necessário aceder o endereço `http://localhost:8080` no navegador.

Não existe um valor certo ou errado para as configurações do arquivo. As alterações devem ser feitas conforme cada caso.

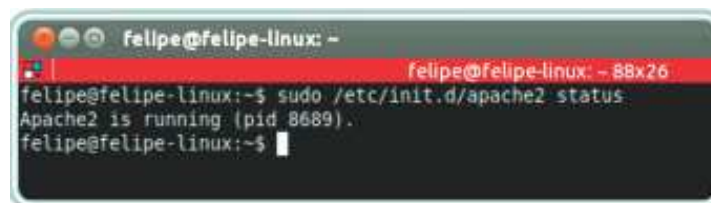
Após o término da edição do arquivo, basta gravá-lo e iniciar ou reiniciar o serviço que o mantém a funcionar. É possível fazer isso executando o seguinte comando: **`sudo /etc/init.d/apache2 start (ou restart)`**

Para saber o estado de um serviço basta utilizar o parâmetro `status`. Por exemplo, para saber o estado do serviço `httpd`, basta executar **`sudo /etc/init.d/apache2 status`**.



```
felipe@felipe-linux: ~$ sudo /etc/init.d/apache2 restart
Restarting web server apache2
apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName
waiting - apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName
felipe@felipe-linux: ~$
```

Figura 22-Reinicialização do serviço `httpd`



```
felipe@felipe-linux: ~$ sudo /etc/init.d/apache2 status
Apache2 is running (pid 8689).
felipe@felipe-linux: ~$
```

Figura 23-Verificação do estado do serviço `httpd`

Após o término da instalação e configuração, o servidor HTTP já estará a funcionar. Para testá-lo, basta aceder o endereço `http://localhost` ou `http://127.0.0.1` através de um navegador qualquer na mesma máquina onde o servidor HTTP está instalado. Esses endereços, quando acedidos, têm como destino o próprio computador. Para aceder o servidor HTTP de outro computador, basta aceder através do endereço IP (*Internet Protocol*) do computador onde o servidor HTTP está instalado. A Figura 24 mostra a página padrão trazida pelo servidor HTTP Apache.

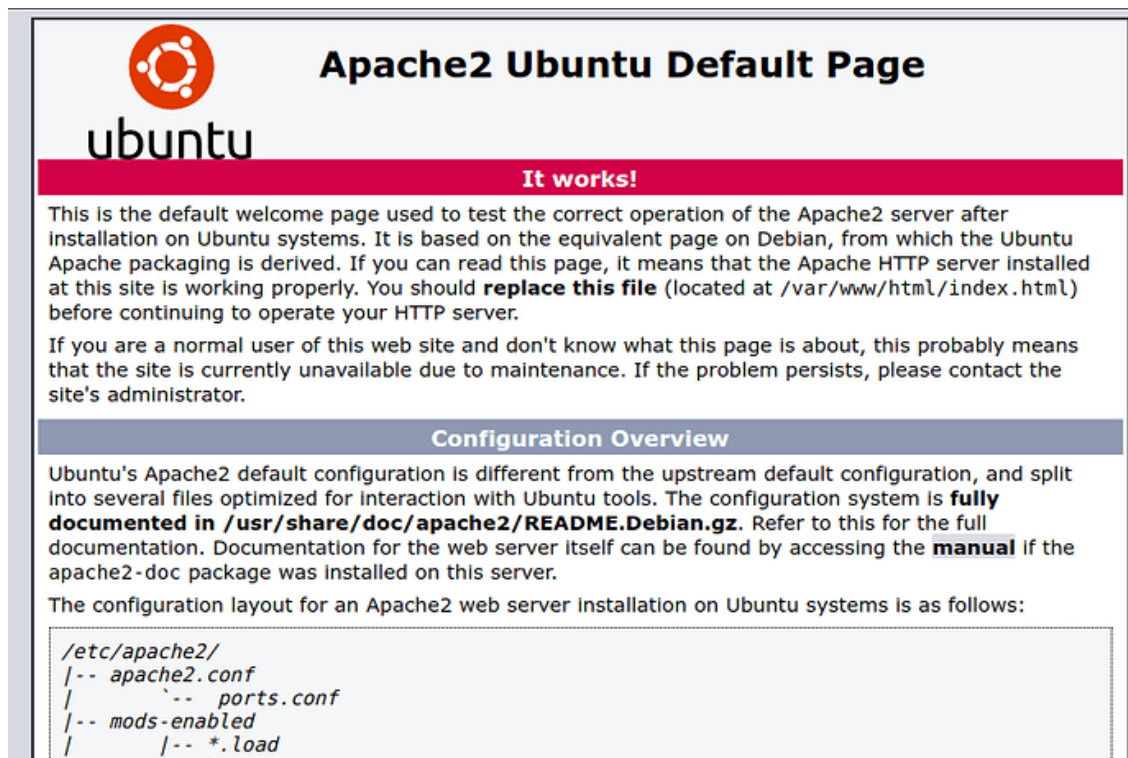


Figura 24-Servidor HTTP a ser acessado pelo navegador

Até o momento, o servidor HTTP pode ser acessado apenas através da rede local, ou seja, apenas em computadores que estiverem na mesma rede (ou na mesma sub-rede de endereços IP) poderá acessar ao servidor. Para que esse servidor possa ser acessado pela internet, a ligação deve ser do tipo empresarial, pois o servidor HTTP usa a porta 80 que, normalmente, é bloqueada pelos provedores de internet. Com uma ligação empresarial e devidamente funcional (velocidade e qualidade adequadas) e com o endereço IP público (fornecido pelo provedor de acesso à internet), é possível acessar o *site* de qualquer lugar do mundo, através do endereço IP. Mas, quando queremos acessar um *site*, escrevemos o endereço num navegador e não o seu endereço IP. Para poder acessar o servidor HTTP através do nome do site, é necessário registrar o endereço do *site* no órgão do país que rege isso. O responsável pela conversão dos endereços de sites para seus respectivos endereços IP é o servidor DNS (*Domain Name System*), que será estudado mais adiante.

Servidor de conteúdo dinâmico (PHP)

Foi mencionado anteriormente, que um servidor HTTP permite que um utilizador requirite uma página de *web* de um *site*, porém ele é capaz de exibir apenas páginas estáticas (em HTML). Isso significa que o servidor não será capaz de interpretar dados enviados pelo utilizador como os usados em formulários, funcionalidades que necessitem autenticação ou armazenamento de informações numa base de dados. Em outras palavras, o servidor não terá condições de exibir conteúdo dinâmico que, no nosso caso, é interpretado pela linguagem PHP, através de arquivos com a extensão PHP.

Para fazer com que o servidor HTTP seja capaz de interpretar e interagir com a linguagem PHP, é necessário a instalação do PHP propriamente dito. Para isso, basta pesquisar os pacotes relacionados ao PHP no gestor de pacotes, através do comando **apt-cache search <nome-do-pacote>**. A Figura 25 mostra os pacotes PHP disponíveis no repositório.

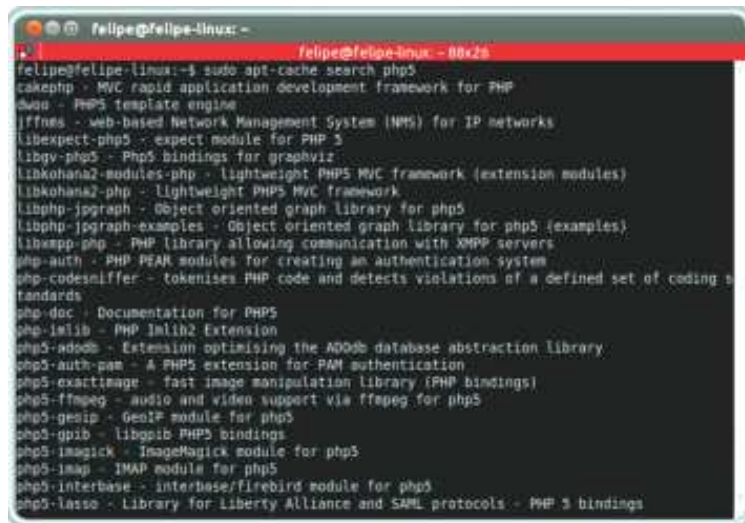
A terminal window titled 'felipe@felipe-linux: -' showing the command 'felipe@felipe-linux:~\$ sudo apt-cache search php5' and its output. The output lists various PHP-related packages and their descriptions, such as 'cakephp - MVC rapid application development framework for PHP', 'twig - PHP5 template engine', 'libexpect-php5 - expect module for PHP 5', 'libgv-php5 - PHP5 bindings for graphviz', 'libkohana2-modules-php - lightweight PHP5 MVC framework (extension modules)', 'libkohana2-php - lightweight PHP5 MVC framework', 'libphp-jpgraph - Object oriented graph library for php5', 'libphp-jpgraph-examples - Object oriented graph library for php5 (examples)', 'libxmp-php - PHP library allowing communication with XMP servers', 'php-auth - PHP PEAK modules for creating an authentication system', 'php-codesniffer - tokenises PHP code and detects violations of a defined set of coding standards', 'php-doc - Documentation for PHP5', 'php-imlib - PHP Imlib2 Extension', 'php5-adoadb - Extension optimising the ADOdb database abstraction library', 'php5-auth-pam - A PHP5 extension for PAM authentication', 'php5-exactimage - fast image manipulation library (PHP bindings)', 'php5-ffmpeg - audio and video support via ffmpeg for php5', 'php5-geosip - GeosIP module for php5', 'php5-gpiib - libgpiib PHP5 bindings', 'php5-imagick - ImageMagick module for php5', 'php5-imap - IMAP module for php5', 'php5-interbase - interbase/firebird module for php5', and 'php5-lasso - Library for Liberty Alliance and SAML protocols - PHP 5 bindings'.

Figura 25-Pesquisa de pacotes para a instalação do PHP

O PHP, ao contrário do Apache (que é o servidor web, ou servidor HTTP) não tem um serviço a ser executado no servidor. Ele traz apenas as ferramentas necessárias para que o servidor *web* consiga manipular as páginas PHP. Assim, não é preciso realizar nenhuma configuração, bastando as configurações que vêm por padrão na instalação.

Para instalar o PHP no nosso servidor Apache, além da instalação do próprio PHP, precisamos de instalar um módulo do Apache que facilita a interação entre o PHP e o Apache: **sudo apt install php libapache2-mod-php**.

Para testar se a configuração do PHP foi realizada corretamente, é necessária a criação de uma página na linguagem PHP para ser executada pelo servidor. Em sistemas operativos Linux, as páginas *web* ficam dentro do diretório **/var/www/**. Após a instalação do servidor HTTP, um arquivo deve ter sido criado dentro desse diretório. Por padrão, esse arquivo chama-se **index.htm** ou **index.html**. Também por padrão, quando o utilizador aceder através de um navegador no próprio servidor os endereços **http://localhost** ou **http://127.0.0.1**, o arquivo exibido é o **index.htm** ou **index.html**. O servidor web é programado para procurar por um arquivo de nome **index**, com as extensões de páginas *web* (**html**, **htm**, **php**, entre outras). Ainda assim, caso o utilizador queira aceder o conteúdo de uma página **html** contida num arquivo com nome diferente, por exemplo, **pagina.html**, basta incluí-la no final dos endereços anteriormente citados que, no caso, seriam **http://localhost/pagina.html** ou **http://127.0.0.1/pagina.html**.

Para realizar o teste do funcionamento do PHP, será criado o arquivo **phpinfo.php** dentro do diretório **/var/www/**. Sua localização completa será **/var/www/phpinfo.php**. Com um editor de texto qualquer (Nano, por exemplo, através do comando **sudo nano /var/www/phpinfo.php**), é necessário inserir o seguinte conteúdo dentro do arquivo.

```
<?php
phpinfo();

?>
```



Figura 26-Edição do arquivo phpinfo.php

Essa estrutura de comandos irá fazer com que, sempre que a página phpinfo.php for acessada, ela exiba as informações da versão do PHP instalada no servidor. Assim, é possível testar o funcionamento do PHP, acessando o endereço <http://localhost/phpinfo.php> ou <http://127.0.0.1/phpinfo.php>. A Figura 27 mostra a página phpinfo.php a ser apresentada no navegador.

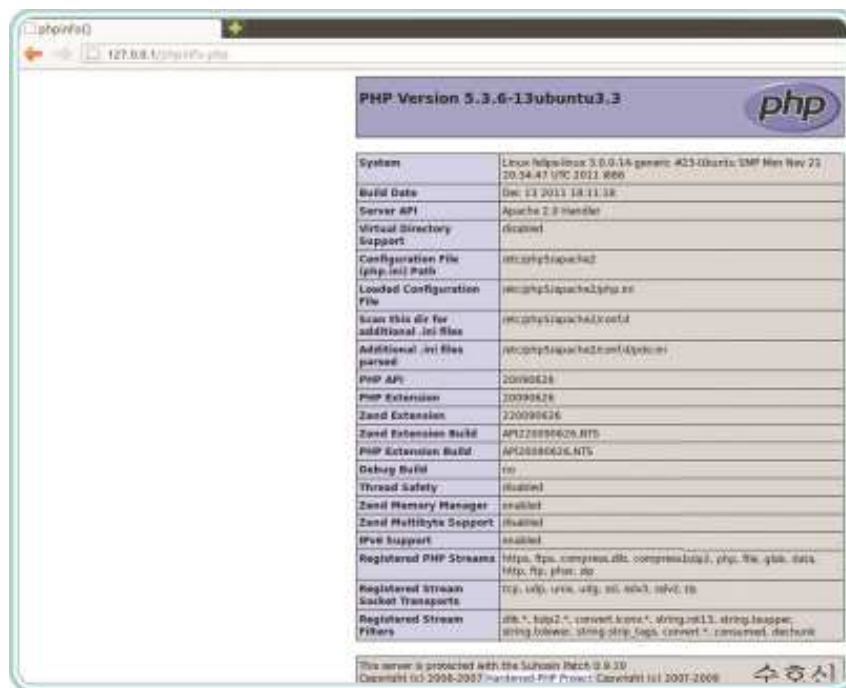


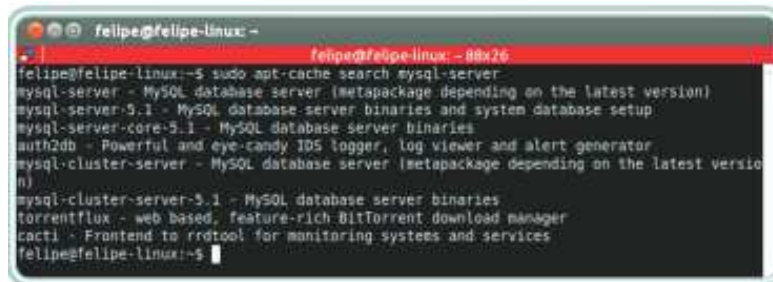
Figura 27-Execução de uma página PHP

Servidor de bases de dados (MySQL)

Para finalizar a configuração de um servidor web básico falta a instalação de um servidor de bases de dados. Uma base de dados nada mais é que um sistema capaz de armazenar informações em tabelas, de forma segura, íntegra e que permita acesso rápido às mesmas. Na grande maioria das vezes, um servidor web não possui a capacidade de interpretar apenas páginas em HTML, trazendo consigo a

capacidade de interpretar conteúdo dinâmico, através do PHP que, por sua vez, precisa armazenar os dados utilizados nos seus formulários, necessitando de uma base de dados. Por exemplo, num site de notícias que são atualizadas várias vezes durante o dia, caso um utilizador queira aceder uma notícia específica, que foi colocada no *site* há vários dias, basta ele pesquisar a notícia. O servidor do *site* irá pesquisar na base de dados, as informações referentes à pesquisa realizada pelo utilizador e irá disponibilizá-las no ecrã, através do PHP.

A instalação do MySQL é realizada da mesma forma que as instalações do servidor HTTP e do PHP através do repositório. A Figura 28 mostra os pacotes do MySQL existentes no repositório, através do comando apt-cache.

A terminal window titled 'felipe@felipe-linux: ~' with a red title bar. The command 'sudo apt-cache search mysql-server' has been executed, resulting in a list of MySQL-related packages. The output is as follows:

```
felipe@felipe-linux:~$ sudo apt-cache search mysql-server
mysql-server - MySQL database server (metapackage depending on the latest version)
mysql-server-5.1 - MySQL database server binaries and system database setup
mysql-server-core-5.1 - MySQL database server binaries
auth2db - Powerful and eye-candy IDS logger, log viewer and alert generator
mysql-cluster-server - MySQL database server (metapackage depending on the latest version)
mysql-cluster-server-5.1 - MySQL database server binaries
torrentflux - web based, feature-rich BitTorrent download manager
cacti - Frontend to rrdtool for monitoring systems and services
felipe@felipe-linux:~$
```

Figura 28-Pacotes do MySQL presentes no repositório

A instalação do MySQL é efetuada através do comando: **sudo install mysql-server**.

Durante a instalação do MySQL, é solicitado ao utilizador, que insira uma senha para o utilizador root da base de dados, ou seja, uma senha de um utilizador com permissões de administrador, capaz de realizar qualquer operação dentro da base de dados. É importante, que esse utilizador seja usado apenas pelo administrador da base de dados ou pelo administrador do servidor, pois com ele é possível apagar todas as informações contidas na base de dados. Por padrão o conjunto de utilizador/palavra-passe na base de dados MySQL é root/root. A linguagem PHP consegue comunicar com a base de dados através de um utilizador e uma senha que devem ser capazes de manipular os dados referentes apenas ao site a que pertencem. Ainda assim, a linguagem PHP presente nos *sites* deve apenas inserir, alterar e excluir informações das tabelas das bases de dados e não devem alterar suas estruturas, pois isso pode gerar problemas nos relacionamentos entre as tabelas. A Figura 29 mostra a tela onde a senha para o utilizador root é solicitada.

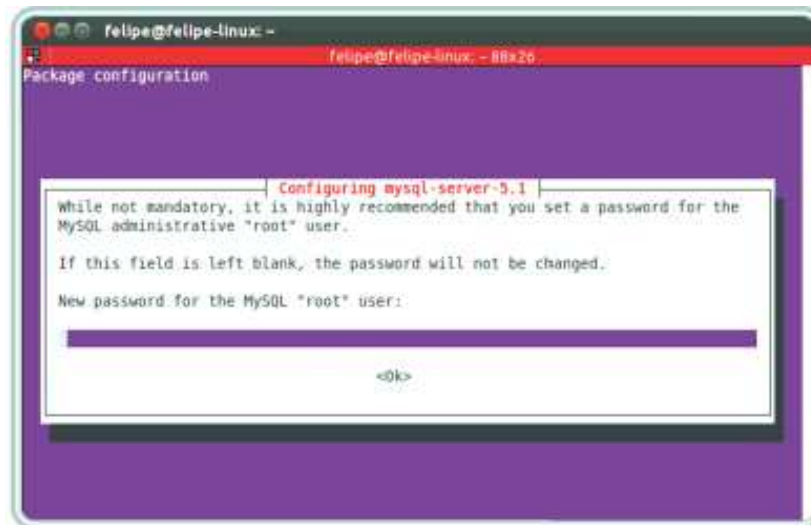


Figura 29-Ecrã da instalação do MySQL a solicitar a criação da senha para o utilizador root

A administração do MySQL fica mais fácil através da aplicação PHPMyAdmin, que pode ser facilmente instalado pelo gestor de pacotes, ou então utilizando o comando: **sudo apt install phpmyadmin php-mbstring php-zip php-gd php-json php-curl**. Após a sua instalação é necessário associar o phpmyadmin ao servidor web:

- `sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf`
- `sudo a2enconf phpmyadmin`
- `sudo systemctl restart apache2`

Por fim, basta aceder através do endereço **http://localhost/phpmyadmin** ou **http://127.0.0.1/phpmyadmin**. Antes, o serviço do MySQL deve ser iniciado, pelo comando **sudo /etc/init.d/mysql start**. A Figura 30 mostra o PHPMyAdmin a ser executado.

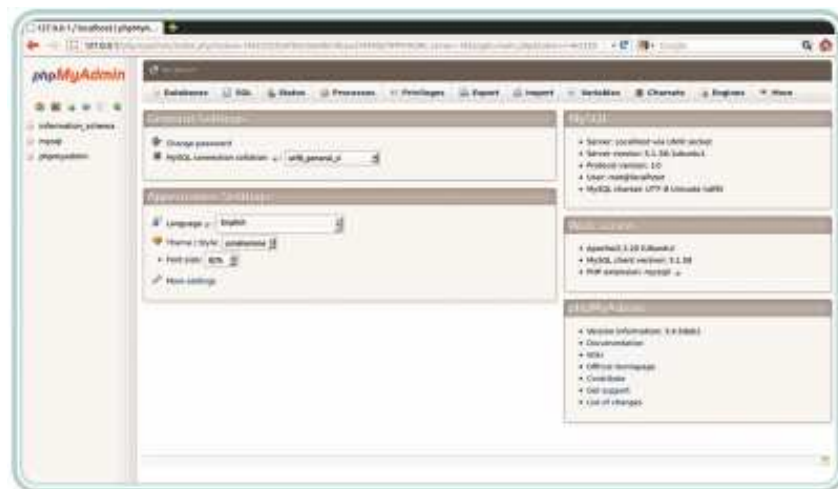


Figura 30-PHPMyAdmin

Configuração do servidor web em ambiente Windows

Até agora foi efetuada a instalação e a configuração em sistemas operativos Linux. Agora vamos efetuar a instalação e configuração mais simples possível de um servidor web em ambiente Windows.

Para isso, vai-se instalar o ambiente XAMPP (pronuncia-se “éksamp”) para que se possa executar código o PHP e testar o código efetuado. O XAMPP é um pacote que traz a linguagem PHP, base de dados MySQL e servidor Web Apache, entre outros serviços, e é *cross-platform*, o que significa que ele pode funcionar em qualquer plataforma, tanto Windows como em Linux. Não será necessário instalá-lo se você decidir usar um dos outros dois ambientes apresentados, porém se quiser experimentar o XAMPP irá conhecer um ambiente muito fácil e configuração simplificada.

Para começar, entre no site oficial do projeto, que é o www.apachefriends.org:



Figura 31-Website do XAMPP

Clique no link de Download para poder descarregar o programa, e na página de download, escolha a versão desejada. Recomenda-se descarregar a versão mais recente. Clique no botão Download da

versão desejada, e espere o programa ser descarregado:



Figura 32-Página de download do XAMPP

Aparecerá a caixa de diálogo de instalação. O instalador recomenda instalar o XAMPP numa pasta que não seja a C:\Program Files, devido a permissões. Clique no botão Next.



Figura 33-Janela inicial de instalação

Na próxima janela poderemos seleccionar os componentes que serão instalados. Poderá deixar todos seleccionados. Clique em Next para prosseguir.

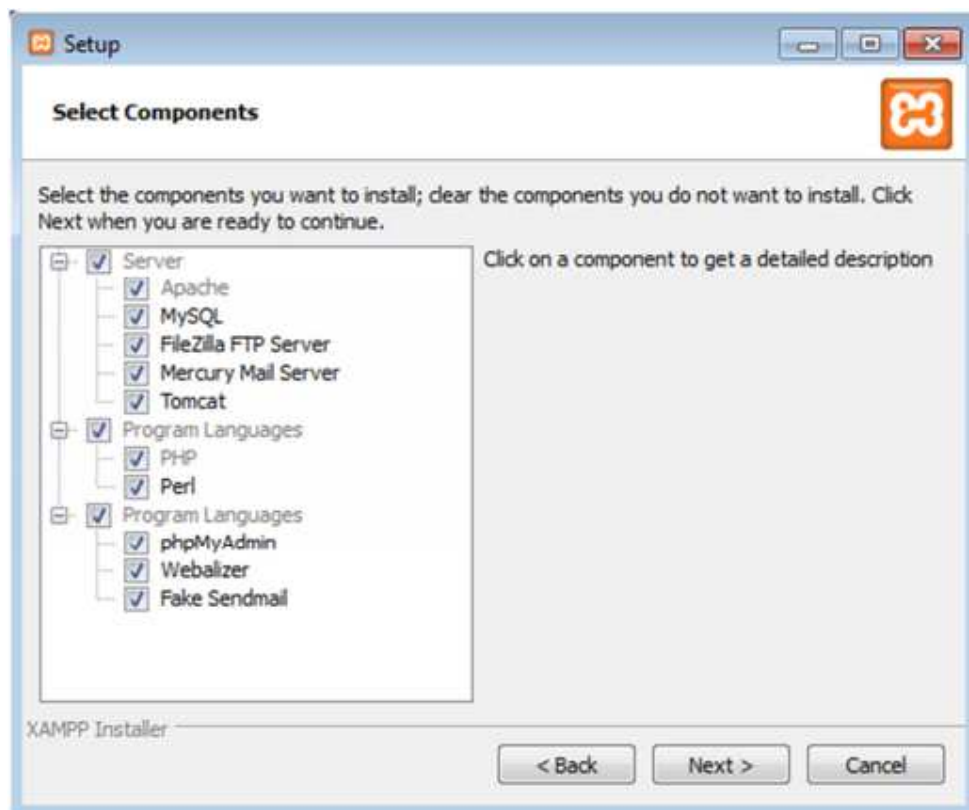


Figura 34-Janela de seleção de componentes a instalar

Na próxima janela, selecione a pasta onde o XAMPP será instalado. Vou instalar o programa na pasta **c:\xampp**.

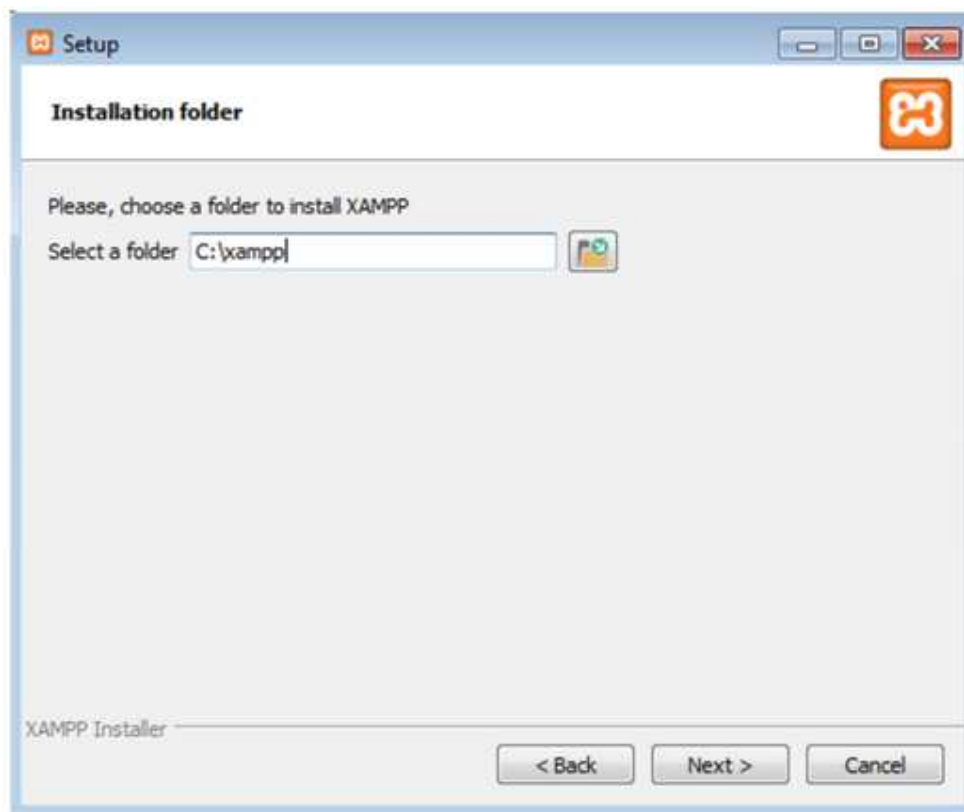


Figura 35-Janela de escolha de diretório de instalação

Pronto para instalar! Clique no botão *Next* para prosseguir.

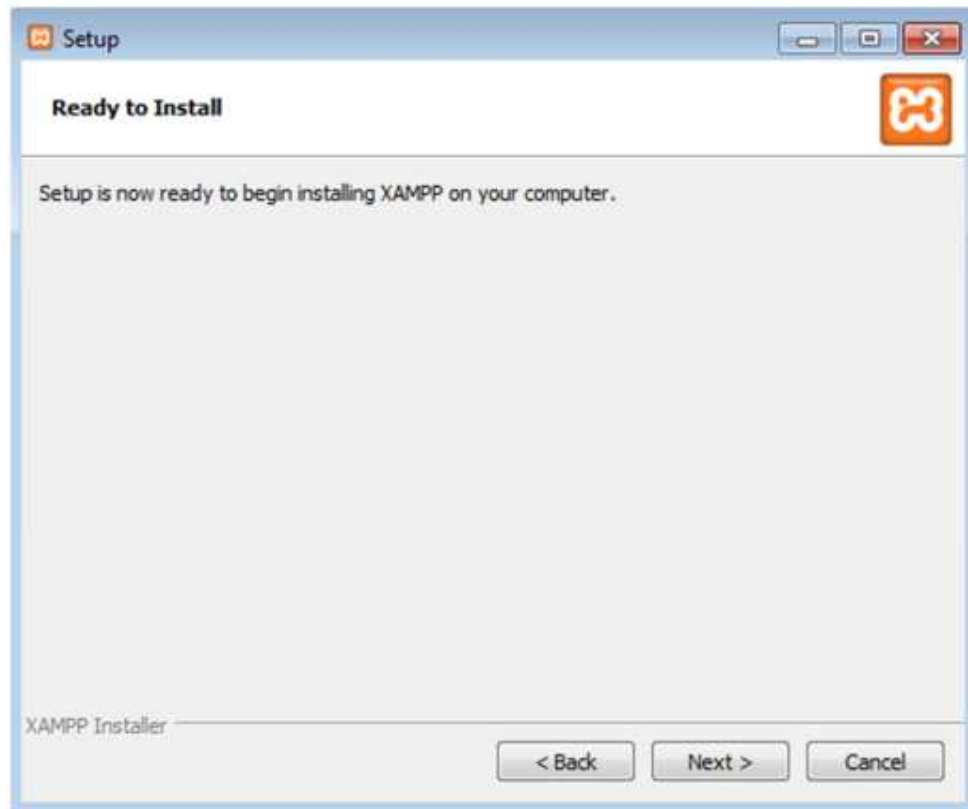


Figura 36-Janela de confirmação de instalação

E aguarde enquanto o XAMPP é instalado em seu computador.



Figura 37-Janela de instalação do XAMPP

Após alguns minutos a instalação será finalizada. Deixe selecionada a caixa **Do you want to start the Control Panel now?** para abrir o Painel de Controlo do XAMPP imediatamente, e clique no botão *Finish* para sair do instalador.



Figura 38-Janela final de instalação

Excelente! XAMPP instalado com sucesso. Veja a janela do Painel de Controlo que se abriu automaticamente. Com ele podemos iniciar, parar e administrar os diversos serviços disponíveis, tais como o servidor Web Apache, base de dados MySQL, serviço de FTP Filezilla, Tomcat e Mercury.

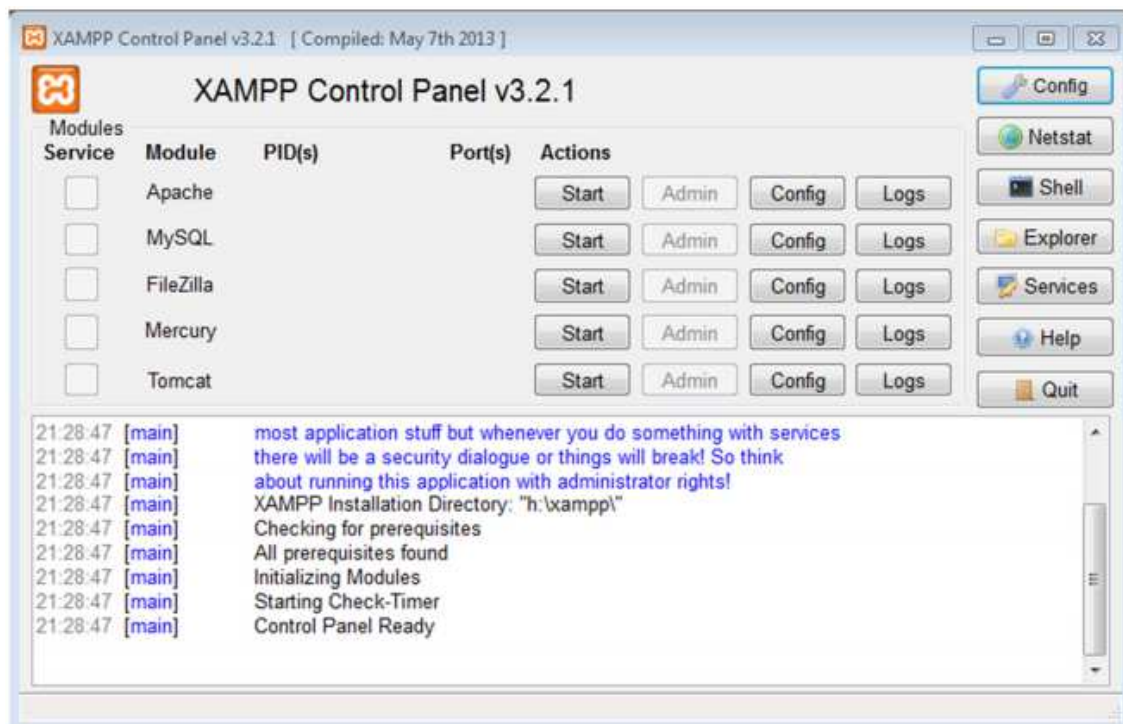


Figura 39-Painel de controle do XAMPP

Clique no botão **Start** do serviço do Apache. De notar que o Alerta de Segurança do Windows poderá ser aberto a perguntar se você quer permitir acesso ao Apache HTTP a partir do Firewall do Windows. Clique no botão **Permitir Acesso**.



Figura 40-Diálogo da firewall do Windows

Faça o mesmo com o serviço do MySQL para iniciá-lo. Veja os serviços ativos no Painel de Controlo do XAMPP.

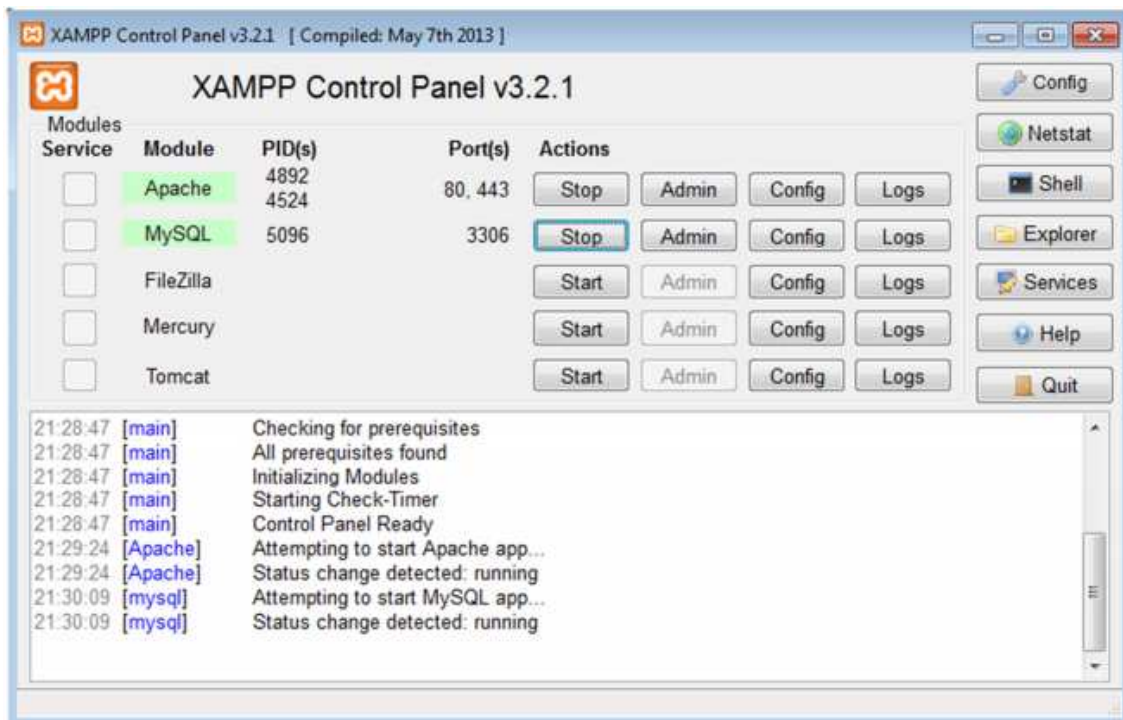
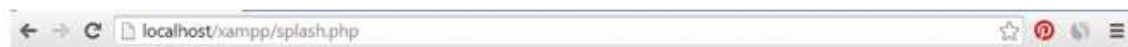


Figura 41-Painel de controlo do XAMPP com serviços ativos

Agora vamos testar a instalação e os serviços que iniciamos no XAMPP. Abra um navegador e escreva o endereço **<http://localhost/xampp>** na barra de endereços. Pressione Enter e verifique se a página a seguir é carregada.



[English](#) / [Deutsch](#) / [Français](#) / [Nederlands](#) / [Polski](#) / [Italiano](#) / [Norwegian](#) / [Español](#) / [中文](#) / [Português \(Brasil\)](#) / [日本語](#)

Figura 42-Janela inicial do XAMPP no navegador

XAMPP instalado e configurado com sucesso! Para começar a usá-lo, abra o Windows Explorer e navegue até a pasta da instalação do XAMPP (no nosso exemplo, `c:\xampp`). Lá dentro encontrará uma pasta de nome **htdocs**, que é onde colocamos os arquivos que iremos. Recomendo que crie uma pasta para cada projeto que executar, dentro dessa pasta **htdocs**. Por exemplo, pode criar uma pasta

chamada projeto1 dentro de htdocs, e criar nesta pasta um arquivo index.php. Para carregar esse arquivo num navegador, basta escrever na barra de endereços: <http://localhost/projeto1/index.php>

Resumo

Foi explicada uma forma fácil e prática de se instalar um servidor HTTP. A instalação das aplicações e as suas bibliotecas através de um gestor de pacotes facilita todo o processo de instalação, pois é rápida e instala automaticamente todas as dependências requisitadas pelas aplicações. Ainda são abordadas as alterações necessárias nos arquivos de configuração das aplicações para fazer com que os mesmos se adequem ao cenário no qual o servidor será instalado. Com o objetivo de mostrar o funcionamento básico de um servidor de um *website*, por exemplo, foram mostradas as instalações do servidor HTTP (através do aplicativo Apache), fazendo com que o servidor seja capaz de responder a pedidos HTTP, ou seja, exibir páginas web (escritas na linguagem HTML), de páginas de conteúdo dinâmico (através da linguagem PHP), permitindo assim que o servidor seja capaz de interpretar envio e recepção de dados através de formulários escritos em PHP e por fim, a utilização de um sistema de gestão de bases de dados (através do MySQL), permitindo que informações enviadas para o servidor possam ser armazenadas nele e, quando solicitadas, apresentadas para o utilizador.

Linguagem PHP

Sintaxe e marcadores de comandos

Vamos falar nesta aula sobre a sintaxe do PHP – a forma correta de escrever o código de forma a ser executado sem problemas.

O código PHP é sempre executado no servidor, e então é gerado HTML que é enviado para o navegador para apresentar as informações processadas.

Podemos escrever os scripts PHP e inseri-los em qualquer parte de um documento, ou então gerar arquivos que contenham somente comandos do PHP, guardando-os em arquivos com a extensão **.php**.

Um script PHP (conjunto de comandos e código) pode ser escrito usando a seguinte sintaxe:

```
<?php  
Códigos PHP;  
?>
```

O código PHP é inserido entre as tags **<?php** e **?>**. Por exemplo, crie um novo documento de texto num editor, insira o código a seguir e guarde-o como `aula01.php`. Recomendo criar um diretório específico para guardar o arquivo. Pode-se usar um diretório de nome `aula01` que deverá estar localizado na pasta adequada, dependendo do pacote de *software* que se vai utilizar para realizar os testes (WAMP, LAMP, XAMPP, etc.):

```
<?php  
echo "Olá Mundo!";  
?>
```

Testar no navegador, abrindo o endereço <http://localhost/aula01/aula01.php>.



Os comandos em PHP terminam sempre com um ponto-e-vírgula (;).

Comentários

Podemos adicionar comentários ao código em PHP, como na maioria das linguagens. O texto que vem após os comentários é sempre ignorado pelo interpretador, e também não aparecerá no navegador ao carregar a página.

Usamos os comentários para que outros programadores possam entender nossa codificação, ou para lembrar a nós mesmos o que fizemos no código, ao editá-lo tempos depois de criado.

// Comentário de linha única.

Outra forma de escrever comentários de linha única

/* Bloco de comentários

que se estende por várias linhas diferentes */

Maiúsculas e minúsculas

No PHP, os nomes de variáveis são *case-sensitive*, ou seja, diferenciam maiúsculas de minúsculas. Portanto, as variáveis **NOME** e **Nome** são diferentes.

Já os nomes de classes, funções e palavras-chave em geral (if, else, while, etc) não diferenciam maiúsculas de minúsculas. mas, de qualquer forma, recomendo criar um padrão para usar.

Tipos de dados

O PHP possui oito tipos de dados disponíveis, divididos em três grupos.

Dados Simples ou Escalares			
Inteiros	Virgula Flutuante	Strings	Booleanos

Dados Compostos	
Objetos	Arrays

Dados Especiais Compostos	
Nulo	Recursos

Tipo Inteiro

Um número inteiro é um número que não apresenta casas decimais, podendo ser positivo ou negativo, e ainda ser representado em bases diferentes, como decimal, octal ou hexadecimal. O formato padrão é o decimal, e o formato octal é representado precedendo o número com um 0, ao passo que o hexadecimal possui os caracteres 0x precedendo o número. Os inteiros são o tipo mais simples em PHP.

O maior inteiro que podemos representar é o valor 2.147.483.647, e o menor (mais negativo) é o número -2.147.483.647, pois os inteiros em PHP são representados por valores de 32 bits.

Alguns exemplos de atribuição de valores inteiros a variáveis:

```
<?php

$a = 15;

$b = -465;

$c = 019;

$d = 0xAF;

var_dump($a);

echo "<br>";

var_dump($b);

echo "<br>";

var_dump($c);

echo "<br>";

var_dump($d);

?>
```

A função **var_dump()** usada no exemplo acima devolve o tipo de uma variável e o seu respetivo valor. Já o código **echo "
";** é usado para enviar o comando HTML **
** (quebra de linha) ao navegador, para que os resultados apareçam um em cada linha.

```
int(15)
int(-465)
int(1)
int(175)
```

Tipo de Vírgula Flutuante

O tipo de vírgula flutuante permite armazenar números que possuam casas decimais, como 23,78 ou 12343,98. Ainda permite usar notação decimal para representar números muito grandes (ou muito pequenos). Veja os exemplos:

```
<?php  
  
$a = 24.65;  
  
$b = 6.02e23;  
  
$c = 1.8E-18;  
  
print("Núm.1: $a <br> Núm.2: $b <br> Núm. 3: $c <br>");  
  
?>
```

Veja que representamos, nas variáveis \$b e \$c, dois números de virgula flutuante usando notação decimal (letras e ou E, indiferentemente). O tamanho máximo de um número de virgula flutuante depende da plataforma onde o programa é executado. O limite é de 1.8e308, usando uma precisão de 14 dígitos decimais (números de 64 bits).

A função **print()** foi usada para mostrar a saída formatada contendo os valores das variáveis.

```
Núm.1: 24.65  
Núm.2: 6.02E+23  
Núm.3: 1.8E-18
```

Tipo Booleano

O tipo de dados booleano aceita apenas dois valores: **True** (verdadeiro) ou **False** (falso).

O tipo booleano não leva em consideração a capitalização das letras, portanto pode-se escrever TRUE, true ou True que não haverá diferenças.

Caso o valor a ser analisado numa expressão seja numérico, será falso se for igual a zero; e verdadeiro se for outro número qualquer.

No caso de uma string, será falso se a string estiver vazia (ou for o caractere 0), e verdadeiro nos outros casos. O mesmo vale para arrays.

E os valores do tipo NULL são sempre falsos.

```
<?php

$x = True;

$y = False;

if ($x==TRUE) {

    echo "Valor verdadeiro <br>";

}

if ($y==TRUE) {

    echo "Valor verdadeiro";

}

else {

    echo "Valor falso";

}

?>
```

No exemplo acima criamos duas variáveis, x e y, e atribuímos a elas os valores True e False, respetivamente. Depois, realizamos um teste condicional lógico com o comando if para testar as variáveis e imprimir os resultados no ecrã com o comando **echo**.

Tipo String

Uma string é uma sequência (cadeia) de caracteres, como uma frase. Não há limites para a quantidade de caracteres que podemos colocar numa variável do tipo string, a não ser a própria memória RAM presente no computador.

Para definirmos uma string, colocamos os caracteres que a compõe entre aspas, que podem ser aspas simples ou plicas. Há diferença na interpretação entre strings com aspas simples ou com aspas duplas, como explicamos a seguir.

Strings com plicas

Uma string definida usando-se plicas é tratada de forma literal, ou seja, o texto presente nela será impresso no ecrã exatamente da forma como foi escrito, incluindo nomes de variáveis, que serão apresentadas no lugar do conteúdo dessas variáveis.

Strings com aspas

Uma string definida com aspas permite mostrar (expandir) o conteúdo das variáveis presentes na string (em vez de somente mostrar seus nomes), e também interpreta determinadas sequências de caracteres (caracteres especiais ou de escape) que são precedidos por uma barra invertida (\).

```

<?php

$nome = 'Fábio';

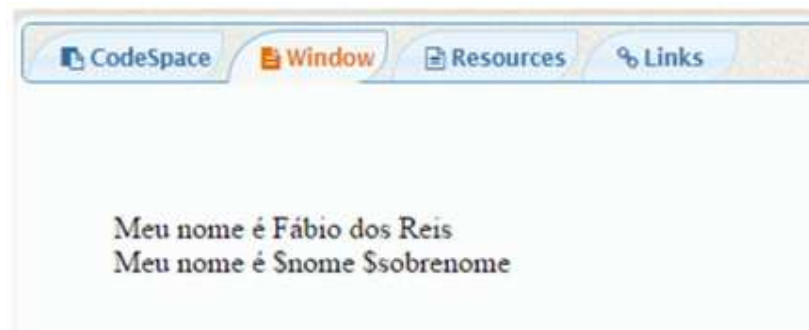
$sobrenome = "dos Reis";

echo "Meu nome é $nome $sobrenome <br>";

echo 'Meu nome é $nome $sobrenome <br>';

?>

```



Na tabela a seguir temos os caracteres de escape (especiais) mais comuns que podem ser usado em PHP:

Caracter	Significado
\n	Linefeed
\r	Carriage Return
\t	Tab
\\	Barra invertida
\"	Aspas
\\$	Cifrão

Função GetType()

A função `gettype()` é um comando que nos permite saber o tipo de uma variável qualquer, retornando uma string com o tipo. Sintaxe exemplo: **`gettype(variável)`**

```

<?php

$num = 10;

$preco = 23.89;

$nome = "Fábio";

$resultado = True;

```

```
echo "$num é " . gettype($num). "\n";

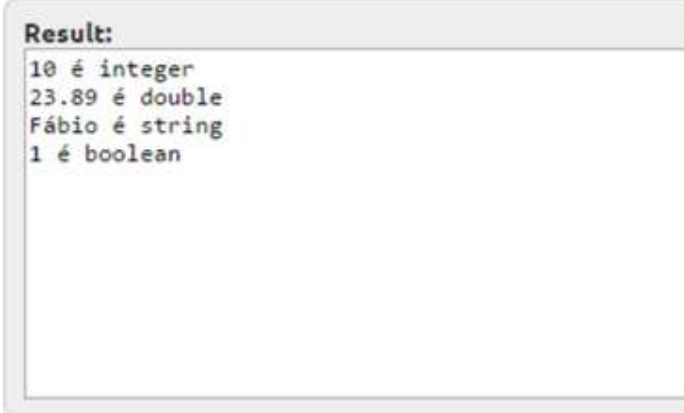
echo "$preco é " . gettype($preco). "\n";

echo "$nome é " . gettype($nome). "\n";

echo "$resultado é " . gettype($resultado). "\n";

?>
```

Tendo como resultado:



```
Result:
10 é integer
23.89 é double
Fábio é string
1 é boolean
```

Note que usamos um ponto (.) entre as strings. Este ponto é um operador de **concatenação**, que permite juntar duas ou mais strings e formar uma nova.

Converter tipos de dados

No PHP podemos converter uma variável de um tipo em outro tipo de três formas distintas:

1. Automaticamente
2. Explicitamente
3. Usando uma função chamada settype()

O PHP pode converter tipos automaticamente em alguns casos, quando ocorrem operações específicas entre dois valores que possuam tipos diferentes.

Para que façamos a conversão de tipos explicitamente usaremos os comandos de cast, de acordo com a tabela a seguir:

Operador	Tipo
int ou integer	inteiro
real, double ou float	real com virgula flutuante
string	string (cadeia de caracteres)
array	vetor
object	objeto

Ver o exemplo a seguir, onde declaramos uma variável do tipo virgula flutuante de nome var1 e depois atribuímos seu conteúdo a uma variável de nome var2, do tipo inteiro. Vamos converter explicitamente o tipo do dado:

```
$var1 = 40.55;  
echo "Ponto flutuante: $var1 <br />";  
$var2 = (int)$var1;  
echo "Inteiro: $var2";
```

Podemos também usar a função `settype()` para efetuar a conversão de tipos no PHP. Essa função permite converter valores nos tipos *integer*, *double*, *string*, *array* ou *object*. Sua sintaxe é a seguinte: **`settype(variável, "novo_tipo");`**

No exemplo a seguir declaramos uma variável do tipo virgula flutuante e a convertemos para inteiro com a função `settype()`:

```
$num = 2.5;  
settype($num,"integer");  
echo "Valor inteiro: $num";
```

Testar o tipo de variável

Podemos testar o tipo de uma variável usando as funções a seguir:

- `is_int()`
- `is_integer()`
- `is_real()`
- `is_long()`
- `is_float()`
- `is_string()`
- `is_array()`
- `is_object()`

Sintaxe: **`is_<TIPO>($variável)`**

Essas funções devolvem o valor verdadeiro se a variável testada for do tipo indicado, e falso em caso contrário.

Um exemplo. Vamos criar uma variável de nome `$num` e atribuir-lhe um valor de virgula flutuante. Depois, testaremos para ver se o valor contido na variável é do tipo inteiro com a função `is_int()`:

```
$num = 23.8;  
if(is_int($num)) {  
    echo "Número inteiro";  
}  
else {  
    echo "O valor da variável não é um número inteiro!";  
}
```

Usamos uma condição `if` para realizar o teste. Se a variável `$num` for do tipo inteiro a função `is_int()` devolverá verdadeiro e então a mensagem Número inteiro será apresentada; caso contrário, a função devolverá falso e a mensagem O valor da variável não é um número inteiro será apresentada.

Declarar e usar variáveis

Uma variável é uma área na memória RAM de um computador que é reservada para armazenar dados de programas em utilização. O seu conteúdo é destruído após a execução do programa ter terminado.

Definimos uma variável no PHP usando a sintaxe a seguir: **\$nome_da_variável = valor_da_variavel;**

O sinal \$ precede sempre o nome escolhido para a variável, tanto na declaração quanto ao atribuímos valores, ou lermos o conteúdo da variável. Logo após esse caracter podemos usar apenas letras ou o sinal de underscore (_). Os outros caracteres devem seguir as regras abaixo:

- Podem ser caracteres alfanuméricos (letras e números, além do *underscore*).
- O primeiro caracter deve obrigatoriamente ser uma letra ou o símbolo de *underscore*.
- Não pode haver espaços em nomes de variáveis compostos – use um *underline* para ligar as palavras ou use a notação CamelCase (palavras unidas, sem espaços, iniciadas com letras maiúsculas)
- As variáveis no PHP são *case-sensitive*, ou seja, fazem distinção entre letras maiúsculas e minúsculas.
- Crie sempre nomes de variáveis significativos, que permitam transmitir uma ideia precisa sobre o conteúdo que a variável armazena.

Vamos declarar uma variável e atribuir um valor. Criaremos a variável nome e atribuiremos a string Fábio. Com o comando *echo* podemos visualizar no browser o valor contido na variável:

```
<?php
$nome = "Fábio";
echo $nome;
?>
```

A variável \$nome foi criada e recebeu uma string como valor. Portanto, ela será do tipo string. Não é necessário especificar o tipo da variável no PHP, pois trata-se de uma linguagem fracamente tipada. O PHP seleciona o tipo da variável baseado no tipo do dado atribuído. Além disso, uma variável pode receber um valor diretamente, como no exemplo anterior, receber uma cópia do conteúdo de outra variável, ou ainda referenciar outra variável (atribuição por referência).

Contexto de variáveis

O contexto de uma variável diz respeito ao alcance dessa variável no programa, o que significa que o contexto define onde a variável é visível ou acessível no *script*. O contexto varia de acordo com o local onde a variável foi criada. As variáveis podem ser **locais** ou **globais**.

Variáveis locais

São as variáveis criadas dentro de funções/ciclos/estruturas condicionais e que só podem ser referenciadas por comandos que se encontram dentro dessa mesmo contexto. Não são válidas fora do contexto onde foram criadas, e são destruídas assim que o contexto deixa de existir.

Variáveis globais

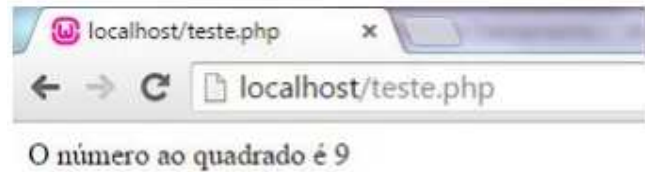
Uma variável global pode ser acedida em qualquer parte do programa, dentro e fora das funções. Para criar uma variável global usamos a palavra reservada **global** antes do nome da variável durante a sua declaração.

Veja o exemplo a seguir, onde criamos uma variável global dentro de uma função e a utilizamos fora da função:


```

<?php
function quadrado() {
    global $numero;
    $numero = $numero * $numero;
}
$numero = 3;
quadrado();
echo "O número ao quadrado é $numero";
?>

```



Variáveis estáticas

Uma variável estática somente existe dentro do contexto de uma função, e não é destruída quando a função é encerrada – o seu valor não é perdido. Podemos utilizá-la novamente ao chamar a mesma função, e ela ainda possuirá o valor que tinha anteriormente. As variáveis estáticas são visíveis apenas no contexto onde foram criadas.

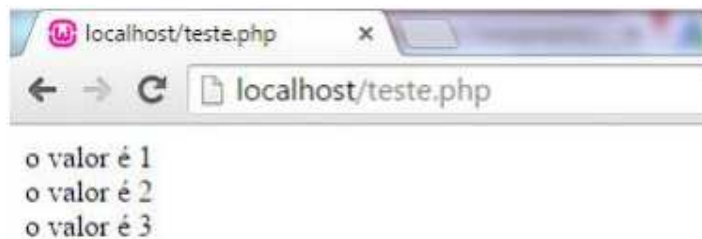
Para criar uma variável estática usamos a palavra chave **static** antes do nome da variável na sua declaração.

Vejamos um exemplo.

```

<?php
function incremento() {
    static $num = 0;
    $num= $num + 1;
    echo "o valor é $num <br />";
}
incremento();
incremento();
incremento();
?>

```



De notar que cada vez que chamamos a função incremento(), o valor apresentado no navegador é incrementado em 1.

Variáveis superglobais

O PHP possui algumas variáveis especiais chamadas de variáveis superglobais que permitem fornecer informações sobre o ambiente do script. Essas variáveis não precisam ser declaradas, pois estão disponíveis automaticamente.

As variáveis superglobais são definidas em arrays, que são coleções especiais de valores. A tabela a seguir mostra as principais variáveis superglobais do PHP:

Nome da variável superglobal	Conteúdo da variável
\$_SERVER	Possui informações sobre o ambiente do servidor web
\$_GET	Possui informações sobre os pedidos GET (formulários)
\$_POST	Possui informações sobre os pedidos POST (formulários)

\$_COOKIE	Possui informações sobre cookies HTTP
\$_FILES	Possui informações sobre uploads de arquivos POST (fotos, documentos, etc...)
\$_ENV	Possui informações sobre ambientes de script
\$_REQUEST	Possui informações sobre pedidos do utilizador
\$_SESSION	Possui informações sobre variáveis registadas na sessão atual
\$GLOBALS	Devolve um array associativo que referencia todas as variáveis disponíveis no contexto global do script
PHP_SELF	Contém o nome do script em execução
DOCUMENT_ROOT	O diretório raiz do script em execução

Um exemplo de uso de variáveis super globais.

```
<?php
    echo "O nome deste arquivo é " . $_SERVER['PHP_SELF'] . "<br/>";
    echo "A pasta onde este arquivo se encontra é " . $_SERVER['DOCUMENT_ROOT'];
?>
```

Neste comando usamos a variável superglobal \$_SERVER e pedimos para mostrar no navegador o conteúdo das variáveis PHP_SELF e DOCUMENT_ROOT, contidas nela.

Variáveis dinâmicas

A linguagem PHP oferece-nos uma categoria muito interessante de variáveis, chamadas de **Variáveis Dinâmicas**, que são variáveis cujos nomes podem mudar, dependendo do conteúdo de outra variável. Para declararmos uma variável dinâmica devemos preceder seu nome com **dois cifrões (\$\$)**, e o PHP irá referenciar o conteúdo dessa variável como sendo uma nova variável.

```
<?php
// Declaramos uma variável chamada $var e atribuímos a string "nome" a ela
$var = "nome";
/* Usando $$ criamos uma variável dinâmica, de modo que o conteúdo de $var seja o nome da
variável (será $nome), e seja atribuído a essa variável a string "Fábio". */
$$var = "Fábio";

// Exibimos no navegador o conteúdo das variáveis $nome e $var:
echo "O nome armazenado é $nome <br/>";
echo "E o conteúdo de var é $var";
?>
```

Operadores

Um operador permite realizar uma transformação sobre um ou mais valores de dados, gerando um novo valor que pode, por exemplo, ser armazenado numa variável, ou ser utilizado como entrada para outras operações.

Os operadores podem ser **unários**, que operam sobre um único valor, **binários**, que operam sobre dois valores ou **expressões** gerando um terceiro valor, e **ternário**, que retorna um valor entre dois possíveis, dependendo do resultado de um terceiro valor ou expressão.

Operadores de atribuição

Os operadores de atribuição são usados quando precisamos atribuir um valor a uma variável. Existe um operador básico de atribuição e vários outros derivados dele. Ver na tabela abaixo os operadores de atribuição que podemos usar em PHP:

Operador	Significado
=	atribuição simples
+=	atribuição com soma
-=	atribuição com subtração
*=	atribuição com multiplicação
/=	atribuição com divisão
%=	atribuição com resto de divisão
.=	atribuição com concatenação

```
$x = 10; //atribui o valor 10 à variável $x.
```

```
$x += 5; //equivale a $x = $x + 5;
```

```
echo "$x <br/>";
```

```
$x /= 2; //equivale a $x = $x / 2;
```

```
echo $x;
```

Os resultados são 15 e 7.5.

Operadores aritméticos

Os operadores aritméticos são usados para realizar cálculos simples, como soma, subtração ou divisão. Devem ser utilizados somente com variáveis de tipos numéricos, ou convertendo-se os tipos com as funções de conversão comuns, caso contrário o PHP tentará fazer uma conversão automática dos tipos.

Na tabela a seguir estão os operadores aritméticos usados no PHP.

Operador	Significado
+	soma
-	subtração
*	multiplicação
/	divisão
%	resto de divisão

O operador de divisão (/) retorna sempre um valor de virgula flutuante, mesmo que os números envolvidos na operação não o sejam.

Além desses operadores temos também os operadores aritméticos unários de incremento e decremento, que podem ser vistos na tabela a seguir:

Operador	Significado
++	incremento
--	decremento

Estes dois operadores podem ser usados antes ou depois das variáveis. Se forem utilizados depois, retornam o valor da variável antes de realizar o incremento ou decremento. Mas se forem usados antes das variáveis, retornam o valor da variável já com o incremento (ou decremento). Alguns exemplos:

```
<?php
    $x = 30; // atribui o valor 30 à variável $x
    $y = $x * 3;
    $z = $y % $x;

    echo "$x <br/>";
    echo "$y <br/>";
    echo "$z <br/>";

    $k = $z++; // atribui a $k o valor de $z e então incrementa $z.

    echo "$k <br/>";

    $k = $z; // atribui a $k o valor de $z, agora já incrementado.

    echo $k;

?>
```

Ordem de Precedência dos Operadores Aritméticos

Na tabela abaixo vemos a ordem de precedência dos operadores aritméticos, da mais alta para a mais baixa (neste caso os de maior precedência estão à esquerda):

Maior precedência			Menor precedência
++,--	- (sinal negativo)	*, /, %	+, -

Podemos subverter a ordem de precedência dos operadores com o uso de parênteses, sendo que as expressões dentro dos parênteses sempre serão avaliadas antes, mesmo que os operadores envolvidos tenham menor prioridade do que os operadores fora dos parênteses.

```
<?php
    $a = 3 + 5 * 4;
    $b = (3 + 5) * 4;

    echo "Valor da expressão sem parênteses: $a <br/>";
    echo "Valor da expressão com parênteses: $b <br/>";

?>
```

Sem o uso de parênteses a expressão devolveria o valor 23 e com parênteses, 32. O valor correto depende da fórmula que se quer avaliar.

Operadores Lógicos, Bitwise e Relacionais

Os operadores **Bitwise**, também conhecidos como operadores bit a bit ou de manipulação de bits, são utilizados para realizar operações em números inteiros que representam valores booleanos (0 e 1), ou em strings caracter a caracter, e podem ser aplicados a qualquer tipo de dados.

Operador	Significado
& ou AND	e
ou OR	ou
~	não
^	xor (ou exclusivo)

Vejamos alguns exemplos de aplicação desses operadores. Vamos usar o operador AND (&) para calcular o resultado da operação bit a bit entre os números 172 e 248. Esse cálculo é usado para determinar o endereço de uma rede, onde uma máscara de sub-rede é aplicada sobre os octetos de um endereço IP, um a um, com a operação e:

```
<?php
$octeto = 172;
$mascara = 248;
$valor = $octeto & $mascara;
echo "Aplicando a operação AND entre a máscara e o octeto obtemos o valor $valor";
?>
```

Operadores de deslocamento de bits

Existem operadores que podem ser utilizados para deslocar bits de um valor para esquerda ou para a direita, o que significa multiplicar número por dois ou dividir o número por dois, respectivamente, a cada deslocamento. Funcionam como multiplicadores e divisores de números.

Operador	Significado
>>	Deslocamento para a direita (divide por 2)
<<	Deslocamento para a esquerda (multiplica por 2)

Por exemplo, $5 \ll 2 = 20$ ($5 * 2 * 2$), e $10 \gg 1 = 5$ ($10 / 2$).

Veja um exemplo do uso de operadores de deslocamento de bits no PHP:

```
<?php
$num = 32;
$valor1 = $num << 2; // equivale a  $32 * 2 * 2 = 128$ 
$valor2 = $num >> 2; // equivale a  $32 / 2 / 2 = 8$ 
echo "$valor1 <br/>";
echo "$valor2";
?>
```

Operadores relacionais

Os operadores relacionais (ou de comparação) permitem efetuar comparações entre valores contidos em variáveis ou constantes entre si, resultando sempre em um valor booleano (lógico, 0 ou 1, verdadeiro ou falso).

Valores diferentes de zero são considerados como valores verdadeiros e o valor zero é considerado como valor falso.

Operador	Significado
==	Igual a
!=	Diferente de
===	Idêntico a (valores iguais e do mesmo tipo)
!==	Não idêntico
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a

```
<?php
$valor = 10;
if ($valor == 15) { // testa se $valor é igual a 15
    echo "Valor é 15";
}
elseif ($valor >= 10) { //verifica se $valor é maior ou igual a 10
    echo "Valor é maior ou igual a 10";
}
else {
    echo "Valor é menor que 10";
}
```

Operador lógicos

Os operadores lógicos permitem realizar comparações entre expressões, devolvendo como resultado um valor verdadeiro (1) ou um valor falso (0). Quando usamos operadores relacionais podemos também combinar as suas operações com operadores lógicos.

Operador	Significado
AND	e
OR	ou
XOR	ou exclusivo
&&	e
	ou
!	negação (not)

```

<?php
$a = 10;
$b = 20;
echo ($a == 10) && ($b == 20);
echo "<br/>";
if (($a == 10) && ($b == 20)) {
    echo "Ambos os valores estão corretos <br/>";
}
if (($a == 20) || ($b == 20)) {
    echo "Um dos valores está correto <br/>";
}
if (($a == 10) XOR ($b == 10)) {
    echo "Um dos valores está errado";
}
?>

```

Os operadores AND e OR são avaliados **ANTES** dos operadores de atribuição; já os operadores && e || são avaliados **DEPOIS** na ordem de prioridade de operadores.

Ordem de precedência dos operadores relacionais

Maior prioridade				Menor prioridade
>, >=, <, <=	==	!=	&&	

Operador de Concatenação de Strings

Concatenar significa unir duas strings (cadeias de caracteres) de modo a formar uma nova string. Existe apenas um operador de concatenação em PHP, que é o ponto (.) e ele, naturalmente, aplica-se somente a operações realizadas com strings.

Se concatenar uma string com um dado de outro tipo, o resultado também será uma string.

```

<?php
$k = "Fábio";
$nome = $k . " dos Reis";
echo $nome;
?>

```

Estruturas de controlo condicional

Uma estrutura de controlo condicional permite controlar o fluxo de execução de um programa, para que seja possível efetuar decisões sobre como executar um determinado comando ou bloco de código de acordo com uma condição, ou executar um bloco de código diferente caso a condição seja diferente.

Controlo Condicional Simples

Permite verificar se uma condição é verdadeira e, em caso afirmativo, executar um comando ou conjunto de comandos associados. Se a condição for falsa, nada é executado.

Se o (teste_condicional) retornar verdadeiro, os comandos serão executados.

```
<?php  
$num = 10;  
if ($num < 20) {  
    echo "O número é menor que 20";  
}  
?>
```

As expressões (teste_condicional) são sempre avaliadas como verdadeiro ou falso (valores booleanos).

Podemos tratar a expressão condicional quando falsa com o uso do condicional composto, que utiliza a palavra reservada **else**.

Controlo condicional composto

Com o controlo condicional composto, podemos ter blocos de código a serem executados tanto para o caso da expressão condicional devolver verdadeiro, como para o caso da expressão condicional devolver falso:

```
<?php  
$num = 30;  
if ($num < 20) {  
    echo "O número é menor que 20";  
}  
else {  
    echo "O número não é menor que 20";  
}  
?>
```

Neste exemplo, alteramos o valor da variável \$num para 30, e acrescentamos o else à estrutura condicional, para que seja possível avaliar o valor se falso.

Podemos também avaliar condições mais complexas, com mais de um teste condicional sequencial, com o uso de **elseif**.

Controlo condicional encadeado

Caso tenhamos a necessidade de avaliar múltiplas condições num programa, podemos fazer uso de um controlo condicional encadeado, que permite encadear várias instruções if. Se o primeiro if devolve verdadeiro, os seus comandos associados são executados, e o controlo condicional é

concluído. Porém, se o if devolve falso, o próximo controlo condicional, dado por elseif será avaliado, e se devolver verdadeiro os seus comandos associados serão executados. Caso devolva falso, o próximo elseif será avaliado, e assim sucessivamente até encontrar um valor verdadeiro, ou executar os comandos associados ao else final, que é obrigatório, e somente são executados quando todos os testes condicionais devolverem o valor falso.

```
<?php  
  
$num = 30;  
  
if ($num < 20) {  
  
    echo "O número é menor que 20";  
  
}  
  
elseif ($num > 20) {  
  
    echo "O número é maior que 20";  
  
}  
  
else {  
  
    echo "O número é exatamente igual a 20";  
  
}  
  
?>
```

Estrutura de controlo switch

A estrutura de controlo **switch** permite substituir uma série de estruturas if encadeadas, testando vários valores para uma mesma variável ou expressão. Desta forma, temos um código mais organizado e fácil de manter e entender.

```
<?php
$diaSemana = 5;
switch ($diaSemana) {
    case 1:
        print ("Domingo");
        break;
    case 2:
        print ("Segunda-feira");
        break;
    case 3:
        print ("Terça-feira");
        break;
    case 4:
        print ("Quarta-feira");
        break;
    case 5:
        print ("Quinta-feira");
        break;
    case 6:
        print ("Sexta-feira");
        break;
    case 7:
        print ("Sábado");
        break;
    default:
        echo "Dia errado informado";
}
?>
```

A variável (ou expressão) será comparada com cada um dos valores nas cláusulas **case** até que seja encontrado um valor que corresponda. Neste caso, os comandos associados ao **case** serão executados, até chegar à cláusula **break**, que encerra a estrutura de controlo.

Se nenhum **case** corresponder ao valor da variável ou expressão testada, serão executados os comandos associados à cláusula **default** (padrão). Esta cláusula é opcional, porém.

A cláusula **break** também é opcional, mas na maioria dos casos irá utilizá-la, pois sem o **break** todos os comandos a partir da correspondência de valores são executados até a última cláusula **case** que contenha um **break**, ou até o final da estrutura **case** – e não apenas os comandos do **case** correspondente à variável ou expressão testada.

Estruturas de repetição

Uma estrutura de repetição permite repetir um comando ou bloco de comandos diversas vezes, automaticamente, de modo a permitir que código seja reutilizado e eliminando a necessidade de reescrever diversas vezes os mesmos comandos que precisam ser repetidos.

Estrutura de repetição while

A estrutura de repetição **while** permite executar repetidas vezes um bloco de comandos enquanto uma condição testada devolve verdadeiro. Quando a condição testada devolve falso, a repetição dos comandos (chamada de loop) é terminada e o programa segue seu fluxo normal de execução.

```

<?php

$num = 1;

while ($num < 31) {

    echo "$num <br/>";

    $num++;

}

?>

```

Nota o uso da instrução `$num++`, que é usada para incrementar o valor da variável testada. Caso não utilizemos essa instrução de incremento, o valor da variável permanecerá sempre em 1 (valor inicial dela), e o programa nunca terminará, pois a condição testada devolverá sempre verdadeiro. Teremos então o famoso “loop infinito”, que é na maioria dos casos um erro de programação.

Estrutura de repetição do ... while

A estrutura **do ... while** difere da estrutura `while` apenas pelo facto de que a estrutura **do .. while** executará sempre o bloco de comandos associado pelo menos uma vez, enquanto a estrutura `while` executa o bloco de comandos zero ou mais vezes – se logo no primeiro teste condicional ele já devolver falso, nenhum comando é executado.

Então, caso precise que o bloco de comandos seja executado pelo menos uma vez numa estrutura de repetição, use o comando **do .. while**.

```

<?php

$num = 10;

while ($num < 10) {

    echo "Executado pelo while: $num <br/>";

    $num++;

}

do {

    echo "Executado pelo do..while: $num <br/>";

} while ($num < 10);

?>

```

Neste exemplo, ao testarmos o valor da variável `$num`, que vale 10, com o comando `while`, temos um valor falso logo de início, e, portanto, o bloco de comandos associado não será executado. Porém, ao testarmos a mesma variável com `do .. while`, o bloco de comandos é executado a primeira vez e só então a expressão será avaliada, devolvendo falso e interrompendo o loop.

Estrutura de repetição for

A estrutura **for** realiza a repetição do bloco de comandos de forma mais refinada do que as estruturas **while**, permitindo que o programador programe a estrutura para que efetue o loop um número pré-definido de vezes, e além disso também permite que se escolha um valor de incremento diferente na própria declaração da estrutura, sem a necessidade de implementar esses parâmetros dentro do bloco de código.

```
for (var_controle; teste_variável; incremento) { <?php  
  
Bloco de comandos                                for ($num = 1; $num <=15; $num++) {  
  
}                                                    print (" $num <br/>");  
  
                                                    }  
  
                                                    ?>
```

Em que:

Var_controle: é uma variável que possui um valor inicial, o qual será avaliado apenas uma vez. É o ponto de partida da contagem de loops a serem realizados. Geralmente inicializamos essa variável dentro da própria estrutura do **for**.

Teste_variável é um teste a ser realizado a cada iteração do loop, e caso devolva falso, o loop é encerrado. Geralmente é um valor limite para a variável de controle, o qual ao ser atingido após incrementos ou decrementos sucessivos, termina o loop.

Incremento é o valor que deve ser adicionado ou subtraído da variável de controle para que ela, após um número determinado de iterações, possa atingir o valor limite que é testado no parâmetro **teste_variável**.

```
<?php  
  
for ($num = 1; $num <=15; $num++) {  
  
    print (" $num <br/>");  
  
}  
  
?>
```

Estrutura de repetição foreach

A estrutura **foreach** permite-nos realizar uma leitura completa num array, devolvendo os seus valores e / ou chaves ou executando operações sobre esses valores. Há duas sintaxes disponíveis para esse comando:

Sintaxe 1: Retornar apenas os valores do array	Sintaxe 2: Retornar os valores e suas respectivas chaves
--	--

foreach (array as variável) { Comandos }	foreach (array as variável-chave => variável-valor) { Comandos }
--	--

Exemplo 1: Criamos um array simples e visualizamos seu conteúdo com o comando **foreach**:

```
<?php

// criando um array para testes:

$vetor = array("SP", "RJ", "PE", "AM", "GO", "SC");

// Visualizando o conteúdo do array:

foreach($vetor as $valores){

    print("$valores <br/>");

}

?>
```

Exemplo 2: Vamos criar outro array e visualizar os seus valores e chaves com o comando **foreach**:

```
<?php

// criando um array para testes:

$vetor = array("SP" => "São Paulo", "RJ" => "Rio de Janeiro", "PE" => "Recife",
"AM" => "Manaus", "GO" => "Goiânia", "SC" => "Florianópolis");

// Visualizando o conteúdo do array e suas chaves:

foreach ($vetor as $estado => $capital) {

    print("A capital de $estado é $capital <br/>");

}

?>
```

Instrução break

A instrução **break** permite interromper a execução dos comandos **for**, **foreach**, **while**, **do..while** e **switch** em qualquer ponto do bloco de comandos. Desta forma, podemos avaliar uma expressão e de acordo com o seu resultado, terminar o loop imediatamente.

```
<?php
for ($x=1; $x <= 30; $x++) {

    print("O valor de x é $x <br/>");

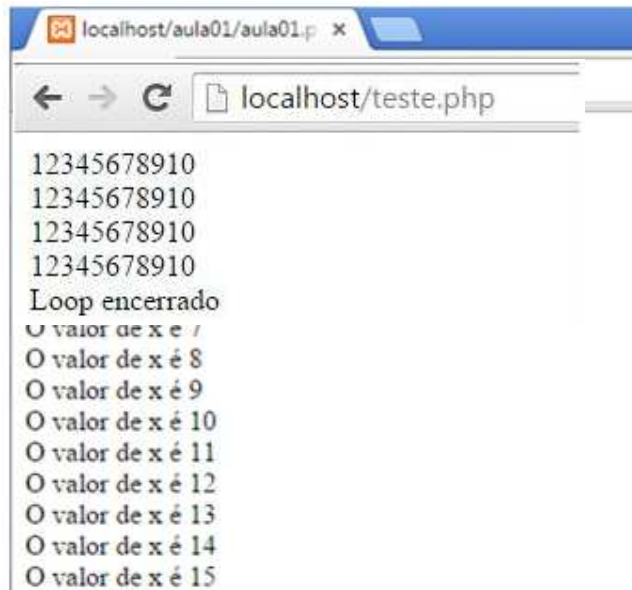
    if ($x == 15) {

        break;

    }

}

?>
```



```
localhost/aula01/aula01.p x
localhost/teste.php
12345678910
12345678910
12345678910
12345678910
Loop encerrado
O valor de x é 7
O valor de x é 8
O valor de x é 9
O valor de x é 10
O valor de x é 11
O valor de x é 12
O valor de x é 13
O valor de x é 14
O valor de x é 15
```

Se houver estruturas de controlo encadeados no código, podemos passar para a instrução **break** um valor que indicará quando deve ser terminado.

```
<?php
for ($y=1; $y <=10; $y++) {
    for ($x=1; $x <= 10; $x++) {
        if ($y == 5) {
            print ("Loop encerrado");
            break 2;
        }
        print($x);
        if ($x % 10 == 0){
            print ("<br/>");
        }

    }

}

?>
```



```
localhost/teste.php
12345678910
12345678910
12345678910
12345678910
Loop encerrado
```

Instrução continue

A instrução **continue**, assim como a instrução **break**, permite que a execução do loop seja interrompida. Porém, diferentemente do que acontece com a instrução **break**, o loop não é terminado – apenas a iteração atual é interrompida, a partir do ponto onde se encontra a instrução, e o loop prossegue na próxima iteração.

```

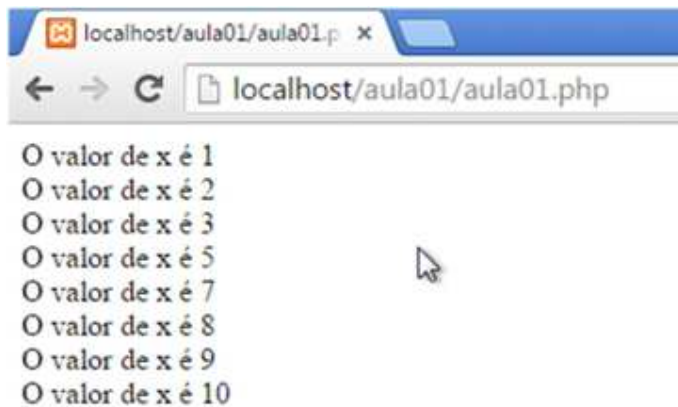
<?php
for ($x=1; $x <= 10; $x++) {

    if (($x == 4) || ($x == 6)) {
        continue;
    }

    print("O valor de x é $x <br/>");
}

?>

```



Declarar e manipular arrays

Um **array** é uma estrutura de dados muito comum, presente em praticamente todas as linguagens de programação. Também conhecido como **vetor**, um **array** pode conter elementos de qualquer tipo no PHP (inteiro, string, ponto flutuante, etc.), o que não acontece com a maioria das linguagens, que exigem que o **array** possua apenas elementos do mesmo tipo.

Podemos criar **arrays** de uma dimensão ou de várias dimensões. Por exemplo, um **array bidimensional** é também conhecido pelo nome de **Matriz**.

No PHP, os elementos de um array podem ser acedidos por meio de índices numéricos (0,1,2,3...) ou ainda por chaves associativas ("nome", "altura", "placa", etc).

Criar um array

Podemos criar arrays usando dois métodos distintos no PHP: com o construtor `Array()` e por criação implícita.

Usando o construtor `Array()`:

```

<?php
// Criando um array vazio:
$vetor = Array();// Criando um array unidimensional:
$vetor2 = Array("Fábio",28,12,150.60);//Criando um array de duas dimensões (matriz):
$matriz = Array(1=>Array("a","b","c"),2=>Array("d","e","f"),3=>Array("g","h","i"));

?>

```

Por meio de criação implícita:

Quando criamos arrays implicitamente, não declaramos o array antes de usá-lo. Assim, o array é criado automaticamente quando referenciado. Usamos neste caso os caracteres `[]` (colchetes) em vez dos parênteses.

```

<?php
$vetor[] = "Fábio";
$vetor2[5] = "Azul";
$vetor3[] = Array(1,2,3);

?>

```

Manipular os elementos de um array

Podemos acessar os elementos de um array usando o nome da variável seguido de parêntesis retos e o índice (ou a chave associativa) do elemento que desejamos acessar. Caso não seja escrito nenhum desses parâmetros, o PHP irá assumir que deseja inserir um novo elemento no array, em vez de acessar a elementos existentes.

Os arrays são indexados a partir do índice 0 (zero), como é comum em muitas linguagens de programação. Assim, um array de 10 posições possui na verdade os índices de 0 a 9.

```
<?php
// Criando um array vazio:
$_vetor = Array();
// Atribuindo um valor ao array criado:
$_vetor[] = "Fábio";
// Visualizando o valor armazenado:
print($_vetor[0] . "<br>");
// Atribuindo mais um valor ao array:
$_vetor[] = "Mônica";
// Atribuindo um valor à posição 4 do array (as posições 2 e 3 terão o valor NULL):
$_vetor[4] = "Renata";
// Visualizando todos os elementos do array com a função print_r():
print_r($_vetor);
// Substituindo um dos valores do array:
$_vetor[1] = "Ana";
// Visualizando todos os elementos do array com a função print_r():
echo "<br>";
print_r($_vetor);
?>
```

Nos exemplos acima, usamos a função **print_r()** do PHP, a qual permite mostrar todos os elementos do array e seus respectivos índices, de uma só vez.

Remover um array ou os seus elementos

Podemos remover um ou mais elementos de um array, ou ainda remover o próprio array com o comando **unset()**. Para remover apenas um elemento, basta informar qual elemento deve ser removido. Caso queira remover o array todo, escrever apenas o nome do array, sem índices.


```
<?php
// Criando um array e atribuindo quatro elementos:
$_vetor = Array("Fábio", "Rita", "João", "Sandra");
// Visualizando todos os elementos do array com a função print_r():
print_r($_vetor);
// Excluindo o elemento 1 do array (Rita):
unset($_vetor[1]);
// Visualizando todos os elementos do array novamente:
echo "<br>";
print_r($_vetor);
// Removendo o array todo:
echo "<br>";
unset($_vetor);
// Visualizando todos os elementos do array (agora haverá um erro):
print_r($_vetor);
?>
```

Processar os dados recebidos de um formulário HTML

Através de um formulário HTML é possível o envio de dados ao servidor, contendo diversos elementos. Após os dados terem sido recebidos pelo servidor, eles devem ser processados, o que significa que podem ser realizados cálculos, apresentados no navegador, armazenados em bases de dados ou então outras ações podem ser disparadas.

Quem irá realizar tais ações é o script PHP que foi referenciado no parâmetro **action** do formulário, contido na tag <form>. Esse script deve estar presente no servidor e irá conter o código PHP necessário para realizar as tarefas desejadas.

Agora vamos ver como criar esse script partindo do formulário seguinte:

```
<!DOCTYPE html>
<html>
<head>
<title>Formulários Web e PHP</title>
</head>
<body>
<form name="cadastro" action="processa.php" method="POST" accept-charset='UTF-8'>
<label>Nome:</label>
<input type="text" name="nome" size="30" placeholder="Digite seu nome aqui" autofocus/><br/>
<label>Sobrenome:</label>
<input type="text" name="sobrenome" size="50" placeholder="Digite seu sobrenome aqui"/><br/>
<br/>
<label>Senha:</label>
<input type="password" name="senha" size=15 required><br/>
<br/>
Sexo: <br/>
<input type="radio" name="sexo" value="M"> Masculino<br/>
<input type="radio" name="sexo" value="F"> Feminino<br/>
<input type="radio" name="sexo" value="N"> Não Declarado<br/>
<br/>
Marque as opções de seu interesse:<br/>
<input type="checkbox" name="linguagens[]" value="ASP"> ASP.NET<br/>
<input type="checkbox" name="linguagens[]" value="JavaScript"> Javascript<br/>
<input type="checkbox" name="linguagens[]" value="PHP"> PHP<br/>
<input type="checkbox" name="linguagens[]" value="Python"> Python<br/>
<br/>
Entre com seus comentários na caixa a seguir:<br/>
<textarea rows="8" cols="50" name="comentarios">
</textarea>
<br/>
Selecione as tecnologias que deseja aprender: (Segure a tecla Ctrl para selecionar mais de uma
tecnologia)<br/>
<select name="tecnologias[]" multiple>
<option value="ASP">ASP.NET</option>
<option value="C" selected>Linguagem C</option>
<option value="C++">C++</option>
<option value="Java">Java</option>
<option value="PHP">PHP</option>
<option value="Python">Python</option>
<option value="Ruby">Ruby</option>
</select><br/>
<br/>
Selecione um arquivo de seu computador para upload:<br/>
<input type="file" name="arquivo"/><br/>
<br/>
<input type="submit" name="submit" value="Enviar" /><br/>
```

O primeiro passo é criar um arquivo no servidor de nome `processa.php`, e guardá-lo no mesmo diretório onde está o arquivo `html` do formulário (pode ser noutro diretório, basta referenciar o caminho corretamente na tag `<form>`, e o nome pode ser diferente também, caso queira).

Array `$_POST` e `$_GET`

Quando os dados são enviados para o servidor, são transformados pelo PHP em elementos de um array associativo de nome `$_POST` ou `$_GET`, de acordo com o método de envio utilizado. As chaves do array são os nomes escolhidos para os campos no formulário, e os seus valores, os dados que os utilizadores inseriram nesses campos. Ou seja, quando o utilizador submete o formulário, os dados que foram inseridos nos campos são armazenados no array (`$_POST` ou `$_GET`), o qual passará esses dados ao script PHP para processamento.

Esses arrays são variáveis **superglobais**, o que significa que estão disponíveis em todos os contextos pelo script, sendo acessíveis dentro de todas as funções e métodos.

Cada elemento dentro do array usado corresponde a uma informação de um campo no formulário. Assim, para aceder o conteúdo de um campo específico, basta usar o nome do array com a chave correspondente. O nome da chave é o mesmo valor do parâmetro “name” usado no formulário para cada campo.

Dessa forma, caso queiramos aceder o conteúdo do campo “Nome” do formulário, cujo parâmetro `name=“nome”`, basta então aceder à chave correspondente no `$_POST`, da forma a seguir:
`$_POST['nome'];`

Com o nome do campo entre plicas, e dentro dos parêntesis rectos. Assim, podemos atribuir esse valor (que contém o dado inserido pelo utilizador) a uma variável, a uma função, ou ainda imprimir no ecrã, ou gravar numa base de dados. Por exemplo, para atribuir esse valor a uma variável chamada “nome” no script: **`$nome = $_POST['nome'];`**

Cuidado para não se confundir: **`$nome`** é uma variável criada no script de processamento, e **`'nome'`** é o nome do campo criado no formulário, que contém a informação transmitida ao servidor, via superglobal `$_POST`.

No caso das *checkboxes* e caixas de seleção, que nos dão a opção de seleccionar mais de um item ao mesmo tempo, precisamos utilizar um **ciclo** para percorrer todo o seu conteúdo e obter os dados presentes. O exemplo abaixo mostra como obter os dados das linguagens de programação seleccionadas pelo utilizador ao clicar nas *checkboxes* correspondentes:

```
// Verificando se os checkboxes foram selecionados
if(isset($_POST['linguagens'])) {

    echo "As linguagens de programação escolhidas foram:<br/>";
    // Efetuar loop pelo array de linguagens:
    foreach($_POST['linguagens'] as $linguagem) {

        echo $linguagem . "<br/>";

    }

}
else {

    echo "Nenhuma linguagem de programação escolhida!<br/>";

}
```

Usamos a função **isset()** para verificar se a variável em questão foi definida – isto é, se **linguagens[]** foi criada quando o formulário foi enviado, devolvendo o valor verdadeiro caso essa variável exista. Caso o utilizador não tenha clicado em nenhuma *checkbox*, **linguagens[]** não será definida e a função **isset()** devolverá falso. Note que não precisamos incluir os parêntesis retos [] ao avaliar a variável com **isset()**.

Se a variável foi definida, então podemos utilizar um ciclo **foreach()** para varrer seu conteúdo (pois **linguagens[]** é um array também) e devolver cada item individualmente. Desta forma:

```
foreach($_POST['linguagens'] as $linguagem) {

    echo $linguagem . "<br/>";

}
```

Assim, devolvemos os elementos individuais presentes em **linguagens[]**, que são as linguagens selecionadas pelo utilizador ao clicar nas *checkboxes*. Usamos a mesma técnica para devolver os dados enviados a partir do campo **SELECT** do formulário:

```

if(isset($_POST['tecnologias'])) {

    echo "Você selecionou as seguintes tecnologias:<br/>";
    // Loop foreach para retornar as tecnologias selecionadas:

    foreach($_POST['tecnologias'] as $tecnologia) {

        echo $tecnologia . "<br/>";

    }

}
else {

    echo "Nenhuma tecnologia selecionada. Que pena! <br/>";

}

```

Se quiser visualizar os dados presentes no array (por exemplo, para fazer *debug* ao script), pode-se usar a função **print_r()**: **print_r(\$_POST)**;

```

Array ( [nome] => Fabio [sobrenome] => dos Reis [senha] => 12345 [sexo] => M [linguagens] =>
Array ( [0] => Javascript ) [comentarios] => Testando o PHP [tecnologias] => Array ( [0] => C [1]
=> C++ ) [submit] => Enviar )

```

Entre parêntesis rectos estão os nomes dos campos do formulário, e associados a eles os seus respetivos valores (informados pelo utilizador). Note que os campos linguagens e tecnologias também são arrays, podendo carregar mais de uma informação em cada um.

Na caixa seguinte pode ver a listagem completa do código do script **processa.php**, usado em nossos exemplos:

```

<!doctype html>
<html lang="pt-br">
<head>
<title>Acesso</title>
</head>
<body>
<?php
$nome = $_POST['nome'];
$sobrenome = $_POST['sobrenome'];
$sexo = $_POST['sexo'];
$comentarios = $_POST['comentarios'];
$senha = $_POST['senha'];
echo "Seu nome é " . $nome . " " . $sobrenome . "<br/>";
echo "O sexo informado foi " . $sexo . "<br/>";
echo "A senha digitada é " . $senha . "<br/>";
echo "Você comentou o seguinte: <br>" . $comentarios . "<br/>";

```

```

if(isset($_POST['linguagens'])) {

    echo "As linguagens de programação escolhidas foram:<br/>";
    // Efetuar loop pelo array de linguagens:
    foreach($_POST['linguagens'] as $linguagem) {

        echo $linguagem . "<br/>";

    }

}
else {

    echo "Nenhuma linguagem de programação escolhida!<br/>";

}

if(isset($_POST['tecnologias'])) {

    echo "Você selecionou as seguintes tecnologias:<br/>";
    // Loop foreach para retornar as tecnologias selecionadas:

    foreach($_POST['tecnologias'] as $tecnologia) {

        echo $tecnologia . "<br/>";

    }

}
else {

    echo "Nenhuma tecnologia selecionada. Que pena! <br/>";

}

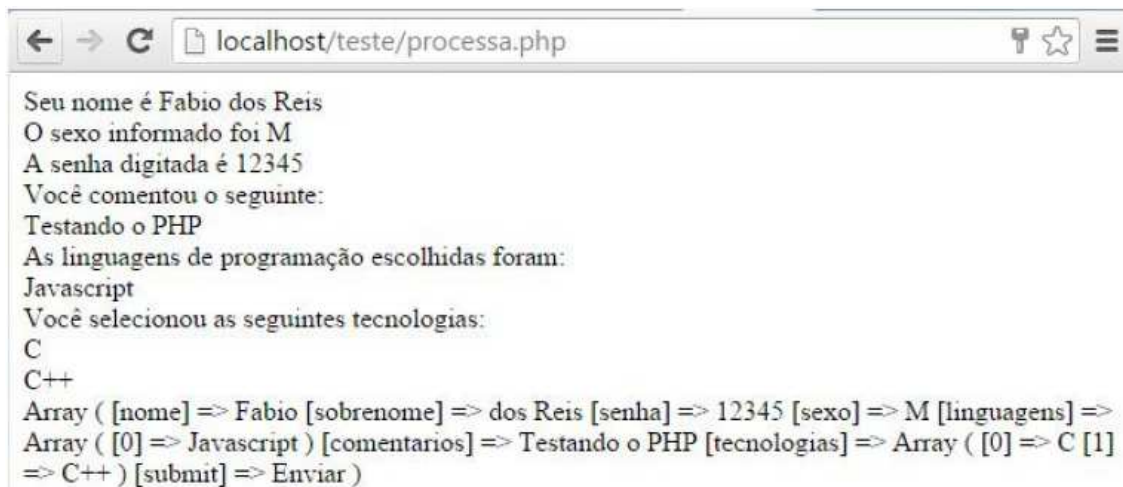
print_r($_POST);

?>

</body>
</html>

```

Após inserir os dados no formulário e clicar em Enviar, os dados são enviados para o servidor e então processados, e o resultado pode ser visto abaixo (nenhum arquivo foi selecionado):



```
Seu nome é Fabio dos Reis
O sexo informado foi M
A senha digitada é 12345
Você comentou o seguinte:
Testando o PHP
As linguagens de programação escolhidas foram:
Javascript
Você selecionou as seguintes tecnologias:
C
C++
Array ( [nome] => Fabio [sobrenome] => dos Reis [senha] => 12345 [sexo] => M [linguagens] =>
Array ( [0] => Javascript ) [comentarios] => Testando o PHP [tecnologias] => Array ( [0] => C [1]
=> C++ ) [submit] => Enviar )
```

[Ligação a uma base de dados MySQL](#)

Vamos agora ver os procedimentos necessários para realizar a ligação do script PHP a uma base de dados MySQL, e executaremos uma inserção de dados para testar a ligação criada.

Vamos usar dois documentos: uma página HTML contendo um formulário de registo de clientes, e um documento PHP contendo as rotinas para tratamento dos dados recebidos da página do formulário.

A seguir temos o código usado para a criação da página com o formulário de registo:

```
<html>
<head>
<title> Testando conexão ao banco de dados </title>
</head>
<body>
<h3>Formulário de Cadastro de Clientes</h3><br>
<form name="Cadastro" action="processa.php" method="POST">
<label>Nome do Cliente: </label><input type="text" name="NomeCliente" size="30"><br>
<label>Sobrenome do Cliente: </label><input type="text" name="SobrenomeCliente" size="45"><br>
<label>Sexo do Cliente: </label><select name="Sexo">
<option value="M">Masculino</option>
<option value="F">Feminino</option>
<option value="N">Não Declarado</option>
</select><br>
<input type="submit" name="enviar" value="Enviar">
</form>
</body>
```

E agora a visualização do formulário criado:

Formulário de Cadastro de Clientes

Nome do Cliente:

Sobrenome do Cliente:

Sexo do Cliente: ▼

Vamos precisar também de uma base de dados criado no MySQL. Vamos criar uma base de dados com o nome **teste**, e um utilizador de nome **fabio**, que será utilizado para efetuar a ligação à base de dados criada. Também iremos dar permissão total a esse utilizador sobre a base de dados. Para tal, ligue-se ao MySQL pela linha de comandos (ou terminal, se estiver no Linux), e execute a sequência de comandos a seguir:

```
CREATE DATABASE banco_teste;
```

```
USE banco_teste;
```

```
CREATE TABLE Cadastro (NomeCliente VARCHAR (20), SobrenomeCliente VARCHAR (30), SEXO CHAR(1));
```

```
CREATE USER fabio@localhost;
```

```
GRANT ALL ON banco_teste.* TO fabio@localhost;
```

```
SET PASSWORD FOR 'fabio'@'localhost' = PASSWORD('123');
```

Dados necessários para a criação do script de procedimento em PHP:

- IP ou hostname do servidor MySQL;
- Nome da base de dados a aceder;
- Nome de utilizador da base de dados;
- Senha do utilizador.

Vejamos agora como efetuar a ligação à base de dados criado. Para isso criar um documento PHP, que guardaremos com o nome de **processa.php**, e então siga os procedimentos a seguir para criar o código deste arquivo.

Efetuar a ligação à base de dados

Passo 1

Primeiramente criaremos uma string de conexão com a função **mysqli_connect()** no arquivo **processa.php**.

A função **mysqli_connect()**: Permite abrir uma ligação a uma base de dados MySQL, de acordo com os dados fornecidos.

Sintaxe: `mysqli_connect('servidor', 'utilizador', 'senha', 'base de dados');`

Essa função devolve uma string de ligação, que deve ser armazenada numa variável para uso posterior, quando formos executar as declarações SQL na base de dados.

Vamos criar a string de ligação e armazená-la numa variável de nome **\$strcon**, usando os dados a seguir:

- Hostname do servidor MySQL: **localhost**
- Nome da base de dados a aceder: **banco_teste**
- Nome de utilizador da base de dados: **fabio**
- Senha do utilizador: **123**

Código:

```
$strcon = mysqli_connect('localhost', 'fabio', '123', 'banco_teste') or die ('Erro ao ligar à base de dados requisitada');
```

Após executar essa instrução, a variável **\$strcon** deve conter os dados necessários para a ligação à base de dados. Usamos a função **die()** que será executada caso haja algum erro durante a execução de **mysqli_connect()**, exibindo a mensagem “Erro ao ligar à base de dados requisitada” como aviso.

Veja como ficará o código no arquivo processa.php para a realização de nosso primeiro teste, que é a ligação à base de dados criada:

```
<?php  
$nome = $_POST['NomeCliente'];  
$sobrenome = $_POST['SobrenomeCliente'];  
$sexo = $_POST['Sexo'];  
$strcon = mysqli_connect('localhost','fabio','123','banco_teste');  
if (!$strcon) {  
die('Não foi possível conectar ao MySQL');  
}  
echo 'Conexão realizada com sucesso!';  
mysqli_close($strcon);  
?>
```

Após criar os documentos (formulário e arquivo processa.php), teste-os abrindo o formulário num navegador, preenchendo os dados e clicando no botão “Enviar”. Se a ligação tiver êxito, verá a mensagem “Ligação realizada com sucesso” na página. Vamos agora modificar o código para que possamos inserir dados na base de dados criada.

Passo 2

Vamos criar a declaração SQL que será executada no servidor, e armazená-la numa variável. Como exemplo, vou inserir um registo novo na tabela **Cadastro** na base de dados (que contém as colunas **NomeCliente**, **SobrenomeCliente** e **Sexo**), a partir de informações recolhidas dos campos nome, sobrenome e sexo de um formulário, armazenando a declaração SQL na variável **\$sql** (o nome pode ser diferente, fica ao critério):

```
$sql = "INSERT INTO Cadastro(NomeCliente, SobrenomeCliente, Sexo) VALUES  
(' $nome', '$sobrenome', '$sexo')";
```

As variáveis **\$nome**, **\$sobrenome** e **\$sexo** contém os dados enviados com o uso do array **\$_POST**.

Passo 3

Executar a declaração SQL na base de dados. Para isso vamos utilizar a função **mysqli_query()**.

Função **mysqli_query()**: Executa uma declaração SQL no banco de dados.

Sintaxe: **mysqli_query(string_ligação, código_SQL, modo_resultado)**

- **string_ligação**: é a string de ligação que foi gerada anteriormente com a função **mysqli_connect()**
- **código_SQL**: é a declaração SQL armazenada na variável **\$sql** criada
- **modo_resultado**:
 - MYSQLI_USE_RESULT (Caso precisemos devolver uma quantidade grande de dados)
 - MYSQLI_STORE_RESULT (Modo padrão)

O **modo_resultado** é opcional.

Exemplo: Vamos agora executar a inserção dos dados obtidos do formulário na base de dados efetivamente:

```
$resultado = mysqli_query($strcon,$sql) or die('Erro ao executar a inserção de dados');
```

A variável **\$resultado** armazenará o resultado da execução da declaração, que pode ser (para o INSERT): TRUE, FALSE ou Failure, dependendo do êxito ou não do processo. É opcional, caso não seja necessário devolver nenhuma informação.

Passo 4

Encerrar a ligação à base de dados. Para isso utilize a função **mysqli_close()**.

Função **mysqli_close()**: Fecha uma ligação a uma base de dados aberta previamente.

Sintaxe: **mysqli_close(string_ligação);**

- **string_ligação**: é a string criada com a função **mysqli_connect()** no início do procedimento.

Exemplo:

Encerrar a ligação usada durante a inserção do registo na base de dados:

```
mysqli_close($strcon);
```

O código completo do script alterado para inserção de dados na base de dados pode ser visto em baixo:

```

<?php
$nome = $_POST['NomeCliente'];
$sobrenome = $_POST['SobrenomeCliente'];
$sexo = $_POST['Sexo'];
$strcon = mysqli_connect('localhost','fabio','123','banco_teste');
if (!$strcon) {
    die('Não foi possível conectar ao MySQL');
}
// Criando a declaração SQL:
$sql = "INSERT INTO Cadastro(NomeCliente, SobrenomeCliente, Sexo)
VALUES ('$nome','$sobrenome','$sexo')";
// Executando a declaração no banco de dados:
$resultado = mysqli_query($strcon,$sql) or die("Erro ao executar a inserção dos dados");
echo "Registro inserido com sucesso";
mysqli_close($strcon);
?>

```

Após a ligação ter sido encerrada, o processo está finalizado e o novo registro do que foi inserido na base de dados com êxito. Efetue uma consulta à base de dados (pelo terminal ou usando o phpMyAdmin), usando a declaração **SELECT * FROM Cadastro**, para constatar que o registro foi incluído com sucesso. Veja o resultado da execução do script após o cadastro do registro "Fabio dos Reis" usando o formulário criado:



```

mysql> SELECT * FROM Cadastro;
+-----+-----+-----+
| NomeCliente | SobrenomeCliente | Sexo |
+-----+-----+-----+
| Fabio       | dos Reis        | M    |
+-----+-----+-----+
1 row in set (0.00 sec)

```

Função die()

Essa função é executada caso haja algum problema na execução das funções mostradas anteriormente, exibindo uma mensagem de erro, e finalizando o script PHP naquele ponto. Passe como parâmetro para esta função a mensagem a ser exibida.

Inserir dados na base de dados MySQL

Trata-se de uma operação extremamente comum e importante, e nós vamos inserir na base de dados provenientes de um formulário HTML.

Usando a nossa base de dados de exemplo anterior, vamos efetuar o registro de dados a partir dos valores inseridos no Formulário de Registro de Clientes, cujos dados serão inseridos na tabela **cadastro** na base de dados **banco_teste**, que possui os campos NomeCliente, SobrenomeCliente e Sexo.

```
mysql> SHOW COLUMNS FROM cadastro;
+-----+-----+-----+-----+-----+-----+
| Field           | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| NomeCliente     | varchar(20)   | YES  |     | NULL    |       |
| SobrenomeCliente | varchar(30)   | YES  |     | NULL    |       |
| Sexo            | char(1)       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Usando o seguinte comando SQL:

```
INSERT INTO cadastro (NomeCliente, SobrenomeCliente, Sexo) VALUES '$nome', '$sobrenome', '$sexo';
```

Onde **\$nome**, **\$sobrenome** e **\$sexo** são os valores obtidos a partir do formulário de registo da página de exemplo:

Formulário de Cadastro de Clientes

Nome do Cliente:

Sobrenome do Cliente:

Sexo do Cliente:

A seguir temos o código da página que contém o formulário de registo de clientes, conforme exibida na figura anterior:

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> Testando conexão ao banco de dados </title>
</head>
<body>
  <h3>Formulário de Cadastro de Clientes</h3><br>
  <form name="Cadastro" action="cadastrar.php" method="POST">
    <label>Nome do Cliente: </label>
    <input type="text" name="NomeCliente" size="30"><br>
    <label>Sobrenome do Cliente: </label>
    <input type="text" name="SobrenomeCliente" size="45"><br>
    <label>Sexo do Cliente: </label>
    <select name="Sexo">
      <option value="M">Masculino</option>
      <option value="F">Feminino</option>
      <option value="N">Não Declarado</option>
    </select><br>
    <input type="submit" name="enviar" value="Enviar">
  </form>
</body>
</html>

```

Ao clicar no botão **Enviar**, os dados são enviados para processamento por um script PHP que se chama `cadastrar.php` para execução do registro.

Código do script PHP `cadastrar.php`

```

<?php
$nome = $_POST['NomeCliente'];
$sobrenome = $_POST['SobrenomeCliente'];
$sexo = $_POST['Sexo'];

$strcon = mysqli_connect('localhost','fabio','123','banco_teste') or die('Erro ao conectar ao banco
de dados');
$sql = "INSERT INTO cadastro VALUES ";
$sql .= "('$nome', '$sobrenome', '$sexo')";
mysqli_query($strcon,$sql) or die("Erro ao tentar cadastrar registro");
mysqli_close($strcon);
echo "Cliente cadastrado com sucesso!";
?>

```

Neste script:

- Obtemos os dados provenientes do formulário HTML por meio da variável `$_POST`
- Abrimos a ligação com a base de dados usando `mysqli_connect()`
- Criamos a declaração sql
- Executamos o comando sql na base de dados usando `mysqli_query()`
- Fechamos a ligação com `mysqli_close()`

Vamos incluir um registo na base de dados a partir de nosso formulário. Veja o form da página preenchido a seguir:

Nome do Cliente:

Sobrenome do Cliente:

Sexo do Cliente:

Após preencher o registo e clicar no botão enviar, os dados serão enviados do formulário para o script **cadastrar.php**, e se não houver nenhum problema, serão gravados na base de dados, e a seguinte mensagem será mostrada na página:



Efetuar uma consulta SQL na base de dados e devolver dados à página HTML

Nas lições anteriores aprendeu-se a criar uma string de ligação e a executar um código SQL na base de dados. Também se mostrou como inserir dados na base de dados, registrando um cliente a partir do preenchimento de um formulário HTML.

Vai-se agora realizar uma consulta à base de dados para que seja possível verificar se os dados foram inseridos corretamente. A consulta deve ser inserida no script PHP de processamento dos dados.

Primeiro, executa-se uma consulta que devolva todos os registos armazenados na base de dados. Vamos usar o formulário a seguir, que contém apenas um botão de consulta:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title> Testando conexão ao banco de dados </title>
</head>
<body>
  <h3>Formulário de Consulta de Clientes Completo</h3><br>
  <form name="Cadastro" action="consulta.php" method="POST">
    <label>Consulta a Banco de Dados completo:</label>
    <input type="submit" name="consulta-completa" value="Consultar">
  </form>
</body>
```

Criando e executando a consulta: **consulta.php**

Vai-se utilizar a função **mysqli_fetch_array()**, que permite obter os resultados de uma consulta SQL e colocar os dados devolvidos num array.

Uso:

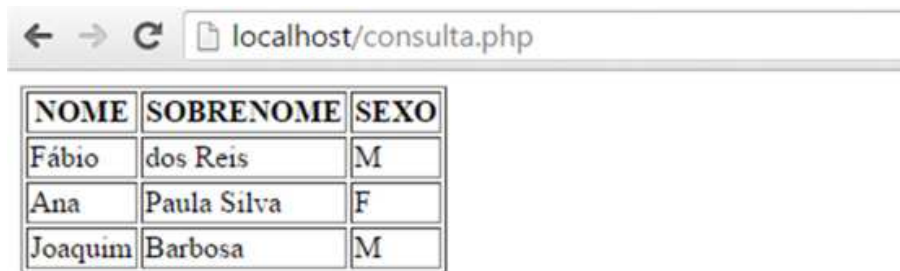
\$registro = mysqli_fetch_array(\$resultado);

A variável **\$resultado** que é utilizada para armazenar o resultado da função **mysqli_query()** não contém dados; ela contém na verdade um número de identificação de um recurso MySQL, que pode ser utilizado para aceder os resultados de uma consulta SQL com a função **mysqli_fetch_array()**, uma linha (registro) de cada vez.

Já a variável **\$registro** é um array que armazena a primeira linha do resultado da consulta. Cada vez que a instrução é executada, uma linha de dados dos resultados da consulta é armazenada no array **\$registro**.

Para obter todos os resultados, basta executar repetidamente a instrução, em ciclo, até que todos os dados tenham sido devolvidos.

Serão apresentados na página os registos armazenados na tabela da base de dados, em cada campo:



NOME	SOBRENOME	SEXO
Fábio	dos Reis	M
Ana	Paula Silva	F
Joaquim	Barbosa	M

Em baixo pode-se ver o script completo do arquivo **consulta.php**, que recebe os dados a serem consultados de um formulário e executa a consulta requisitada:

```
<?php
// Criando tabela e cabeçalho de dados:
echo "<table border=1>";
echo "<tr>";
echo "<th>NOME</th>";
echo "<th>SOBRENOME</th>";
echo "<th>SEXO</th>";
echo "</tr>";
// Conectando ao banco de dados:
```



```
$strcon = mysqli_connect('localhost','fabio','123','banco_teste') or die('Erro ao conectar ao banco de dados');  
$sql = "SELECT * FROM cadastro";  
$resultado = mysqli_query($strcon,$sql) or die("Erro ao retornar dados");  
// Obtendo os dados por meio de um loop while  
while ($registro = mysqli_fetch_array($resultado))  
{  
    $nome = $registro['NomeCliente'];  
    $sobrenome = $registro['SobrenomeCliente'];  
    $sexo = $registro['Sexo'];  
    echo "<tr>";  
    echo "<td>".$nome."</td>";  
    echo "<td>".$sobrenome."</td>";  
    echo "<td>".$sexo."</td>";  
    echo "</tr>";  
}  
mysqli_close($strcon);  
echo "</table>";  
?>
```


Servidores FTP

Objetivos

Instalar, configurar e gerir um servidor FTP (File Transfer Protocol), bem como conhecer o seu funcionamento.

Considerações iniciais

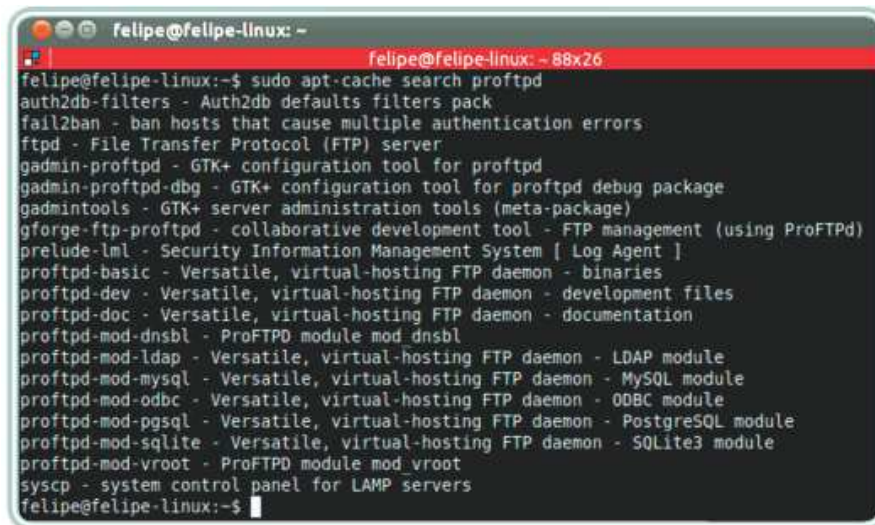
Um servidor FTP tem como objetivo disponibilizar arquivos para os utilizadores. Um exemplo muito comum são os repositórios de sistemas Linux, pois eles trazem uma enorme quantidade de arquivos que ficam acessíveis a todos os utilizadores do sistema. Outro caso onde é muito comum o uso de servidores FTP é em servidores web, pois as páginas de Internet ficam em constante desenvolvimento e a cada alteração o arquivo presente no servidor onde o site está, precisa ser substituído pela versão mais atual. Assim, é necessário enviar esses arquivos para o servidor. Instalando um servidor FTP, juntamente com o servidor HTTP, facilita-se a transferência desses arquivos. Aplica-se também em empresas que desenvolvem sistemas Linux que, ao disponibilizar uma nova versão do sistema, permitem que os utilizadores descarreguem a imagem do sistema através de FTP.

Instalação

A instalação de um servidor FTP é tão simples quanto a instalação do dnsmasq, pois consiste apenas na pesquisa pelos nomes dos pacotes corretos no repositório da distribuição Linux e, posteriormente, na sua instalação, através do gestor de pacotes.

Existem atualmente vários servidores FTP para Linux, mas será abordado apenas o proftpd. Entretanto, a instalação e configuração de outras aplicações de servidor FTP seguem o mesmo padrão, de forma que as configurações a serem alteradas são bastante semelhantes.

Para realizar a instalação do proftpd, basta pesquisarmos pacotes disponíveis no repositório do Linux, conforme mostra a Figura 43.



```
felipe@felipe-linux: ~  
felipe@felipe-linux: - 88x26  
felipe@felipe-linux:~$ sudo apt-cache search proftpd  
auth2db-filters - Auth2db defaults filters pack  
fail2ban - ban hosts that cause multiple authentication errors  
ftpd - File Transfer Protocol (FTP) server  
gadmin-proftpd - GTK+ configuration tool for proftpd  
gadmin-proftpd-dbg - GTK+ configuration tool for proftpd debug package  
gadmintools - GTK+ server administration tools (meta-package)  
gforge-ftp-proftpd - collaborative development tool - FTP management (using ProFTPd)  
prelude-lml - Security Information Management System [ Log Agent ]  
proftpd-basic - Versatile, virtual-hosting FTP daemon - binaries  
proftpd-dev - Versatile, virtual-hosting FTP daemon - development files  
proftpd-doc - Versatile, virtual-hosting FTP daemon - documentation  
proftpd-mod-dnsbl - ProFTPD module mod_dnsbl  
proftpd-mod-ldap - Versatile, virtual-hosting FTP daemon - LDAP module  
proftpd-mod-mysql - Versatile, virtual-hosting FTP daemon - MySQL module  
proftpd-mod-odbc - Versatile, virtual-hosting FTP daemon - ODBC module  
proftpd-mod-pgsql - Versatile, virtual-hosting FTP daemon - PostgreSQL module  
proftpd-mod-sqlite - Versatile, virtual-hosting FTP daemon - SQLite3 module  
proftpd-mod-vroot - ProFTPD module mod_vroot  
syscp - system control panel for LAMP servers  
felipe@felipe-linux:~$
```

Figura 43-Pacotes do proftpd disponíveis no repositório do Ubuntu

Configuração

Após realizar a instalação, é necessário alterar os parâmetros de configuração presentes no arquivo /etc/proftpd/proftpd.conf. Para isso, basta editar o arquivo com um editor de texto qualquer, através

do comando `sudo nano/etc/proftpd/proftpd.conf`, por exemplo. A Figura 44 mostra os parâmetros mais importantes e sua descrição.

Parâmetro	Descrição
DisplayConnect	Permite a edição de um arquivo que exibirá uma mensagem personalizada, o arquivo é <code>/usr/local/etc/proftpd.banner</code> .
ServerIdent	Exibe ou não informações sobre o sistema no qual o servidor FTP está rodando.
Port	Porta na qual o servidor irá operar (a porta padrão é a 21).
MaxClients	Número máximo de clientes conectados simultaneamente.
MaxClientsPerHost	Número máximo de clientes por computador conectados simultaneamente.
DefaultServer	Indica se o servidor FTP é o principal ou não.
UserAlias	Usuários que terão permissão de acessar o servidor, através de <i>login</i> e senha.
ServerName	Nome do servidor.
RootLogin	Habilita ou não o uso do usuário root. Por questões de segurança, a melhor opção é não permitir o uso do usuário root.
User	Usuário no qual o servidor normalmente deve operar.
Group	Grupo no qual o servidor normalmente deve operar.
DefaultRoot	Indica se o usuário poderá sair de sua pasta pessoal depois de conectado ao servidor. Por questões de segurança, a melhor opção é não permitir que os usuários saiam de sua pasta pessoal. Para fazer isso, basta adicionar <code>~</code> após o parâmetro.

Figura 44-Parâmetros de configuração do proftpd

Os utilizadores que terão acesso ao servidor FTP são utilizadores reais do sistema operativo do servidor e assim, cada um deles possui uma pasta pessoal no servidor. Portanto, cada utilizador que se ligar, terá acesso apenas à sua pasta, caso a opção `DefaultRoot` esteja definida como `~`. Ainda assim, é possível permitir acesso de utilizadores anónimos. Isso significa que um utilizador poderia aceder ao servidor sem se identificar. É comum, nesse caso, fazer com que utilizadores anónimos não tenham permissões para escrever na pasta onde os arquivos públicos estão. Assim eles só poderão copiar os arquivos que estão na pasta, mas não poderão apagá-los nem alterá-los.

Na instalação, o `proftpd` cria um utilizador chamado `ftp` e cria a pasta pessoal para esse utilizador em `/home/ftp/`. Da mesma forma, cada utilizador que aceder ao servidor autenticando-se, terá sua pasta `/home/nome-do-utilizador`.

Para permitir acesso por utilizadores anónimos, é necessário, no arquivo de configuração, criar uma estrutura indicando que o acesso anónimo será permitido, e indicar a pasta na qual ficarão os arquivos de acesso público.

A Figura 45 mostra o um pouco do arquivo de configuração, devidamente comentado, e como realizar essa configuração.

```

1. <Anonymous /home/ftp      #indica que a pasta será /home/proftpd
2.   User ftp                #indica o usuário a ser utilizado.
3.   Group ftp               #indica o grupo a ser utilizado.
4.   UserAlias anonymous ftp  #permite que os usuários se conectem anonimamente com os
                               usuários anonymous e Proftpd.
5.   MaxClients 10           #número máximo de clientes simultâneos.
6.   <Directory *>          #qualquer diretório dentro da pasta pública.
7.       <Limit WRITE>      #limitação de escrita.
8.           DenyAll        #nega qualquer escrita.
9.       </Limit>
10.   </Directory>
11. </Anonymous>

```

Figura 45-Estruturas para permitir acesso anônimo

Ainda existem outros parâmetros de configuração que podem ser encontrados no manual.

Após a configuração, é necessário reiniciar o servidor, através do comando **sudo /etc/init.d/proftpd restart** ou **sudo service proftpd restart** para que as configurações tenham efeito.

Para aceder o servidor FTP, basta inserir o endereço IP do servidor em um navegador, utilizando o protocolo ftp. Ex.: ftp://127.0.0.1. Caso a porta do servidor tenha sido alterada, é necessário informá-la da seguinte forma: ftp://127.0.0.1:2000, para o caso em que a porta é a 2000. Ao realizar isso, abrir-se-á uma janela solicitando um nome de utilizador e palavra-passe, conforme mostra a Figura 46. Caso sejam inseridos um utilizador e palavra-passe, a página apresentada mostrará os arquivos presentes na pasta do utilizador preenchido, como mostra a Figura 47. Para aceder como anônimo, basta inserir “anonymous” no nome de utilizador e deixar a palavra-passe em branco. A Figura 48 mostra o conteúdo da pasta pública.

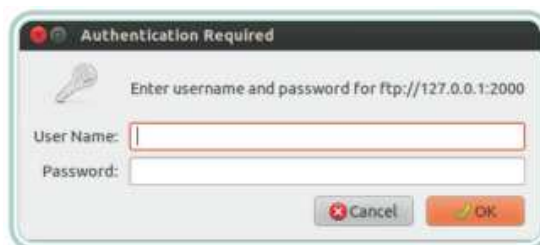


Figura 46-Solicitação de utilizador e palavra-passe

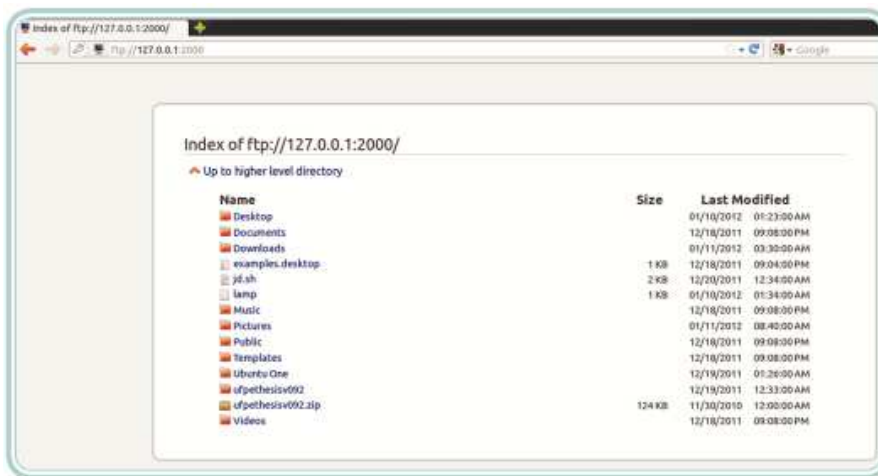


Figura 47-Acesso de um utilizador com autenticação



Figura 48-Acesso anônimo

Entretanto, através de um navegador, é possível apenas visualizar e descarregar os arquivos presentes no servidor, mesmo com um utilizador que tenha permissão para alterar o conteúdo de sua pasta.

Esse problema é facilmente contornado utilizando um cliente FTP como o gFTP ou Filezilla ou algum outro disponível. Existem alguns plugins que são extensões para os navegadores que permitem a utilização de um cliente FTP. Independentemente do cliente utilizado, ele permite que um utilizador, após autenticar-se, possa transferir arquivos para o servidor e copiar os arquivos presentes nele. Caso o cliente aceda o servidor FTP através de um computador com sistema operativo Microsoft Windows, é possível aceder a um servidor FTP, escrevendo o endereço do servidor (ex.: ftp://127.0.0.1:2000) no Windows Explorer. Será solicitado um utilizador e palavra-passe e, após a autenticação, será possível enviar arquivos para o servidor simplesmente copiando-os de sua origem e colando na pasta aberta pelo FTP.

Com o objetivo de melhor visualizar o funcionamento dos processos citados, algumas operações serão mostradas a seguir. A Figura 49 mostra o conteúdo da pasta /home/proftpd, que será a pasta pública do servidor FTP. As Figuras 50 e 51 mostram, respetivamente, o acesso ao servidor através do Microsoft Windows Explorer e o acesso através do cliente Filezilla.

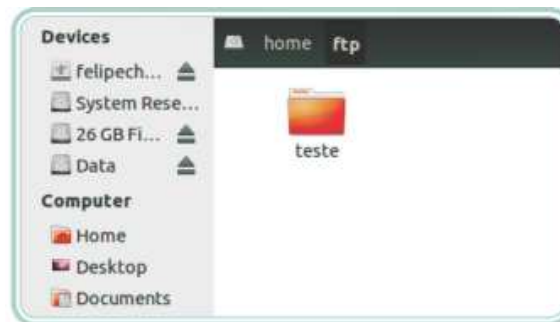


Figura 49-Conteúdo da pasta /home/proftpd

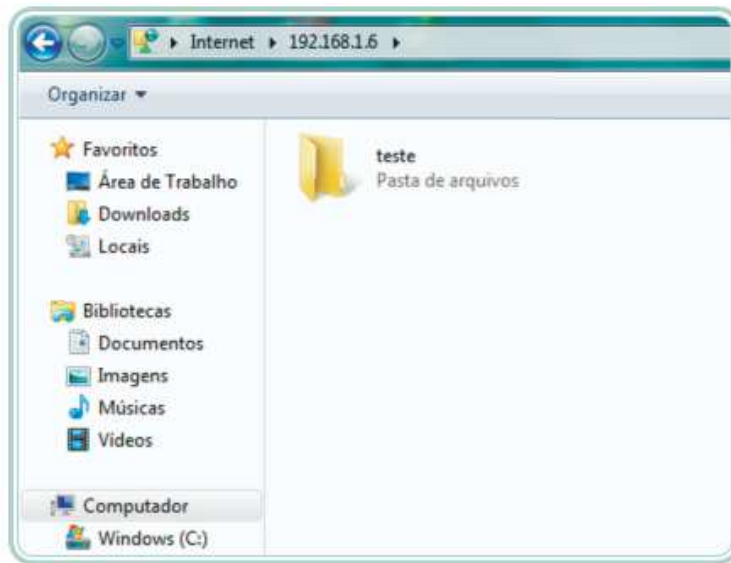


Figura 50-Acesso através do Microsoft Windows Explorer

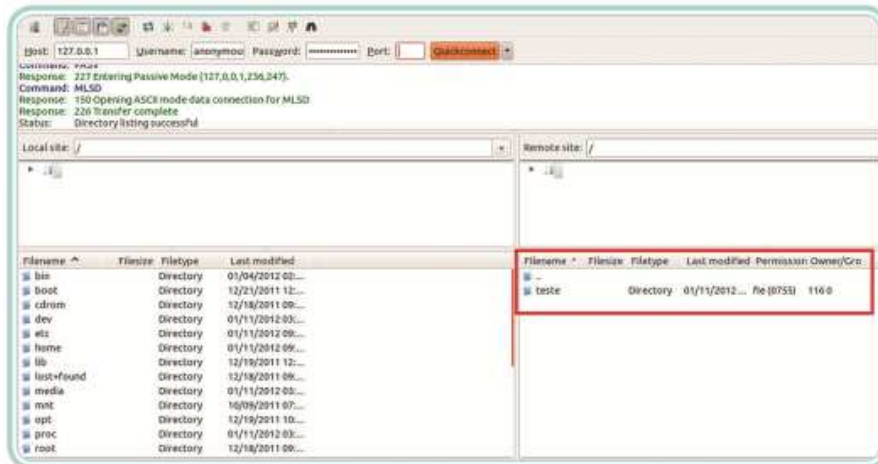


Figura 51-Acesso através doo Filezilla

Resumo

Apresentou-se o conceito de servidores FTP, trazendo a sua utilidade e aplicação. Com foco numa abordagem prática, viu-se um passo a passo para realizar a instalação de um servidor FTP, através de um gestor de pacotes de um sistema GNU/Linux Ubuntu. Após a instalação, o servidor FTP já está operacional, pois na sua instalação, a aplicação responsável por gerir o serviço (proftpd, no caso) traz configurações capazes de tornar o serviço operante. Porém, é necessário personalizar o serviço para se adequar às necessidades do ambiente no qual ele será instalado. Por esse motivo, apresentou-se também um passo a passo sobre os parâmetros de configuração mais importantes do serviço. Por fim, exemplificou-se o acesso ao serviço, mostrando como ele pode ser acedido pelos clientes, deixando clara a finalidade das configurações realizadas.

Servidores DNS

Objetivos

Instalar, configurar e gerir um servidor DNS (*Domain Name System*) e dominar conhecimentos sobre o seu funcionamento e finalidade.

Considerações iniciais

Os servidores DNS estão presentes em todas as redes de computadores que possuem acesso à *Internet*, pois são eles os responsáveis pela tradução dos nomes de endereços de *sites*. Cada vez que uma página da internet é solicitada numa rede, o servidor DNS traduz o endereço do site para o IP do servidor onde ele se encontra.

Ou seja, o servidor DNS traduz os endereços escritos no navegador, como por exemplo `www.w3c.org`, para o endereço IP do servidor web no qual o site está alojado. É muito mais fácil decorar um endereço pelo nome do que por um endereço IP.

O servidor DNS possui uma lista com uma enorme quantidade de endereços de sites e dos seus respetivos endereços IP. Assim, quando um *site* for solicitado por um utilizador, essa lista é verificada, e o endereço do site é acedido. Caso o servidor DNS não possua o endereço IP do site requisitado, ele mandará para o utilizador uma página informando que não conseguiu encontrar a website ou mandará esta solicitação para outro servidor DNS que, por sua vez, caso não tenha, pode responder da mesma forma, informando que não conseguiu encontrar a página ou enviar a solicitação para outro servidor e mandar a resposta correta, caso o outro servidor a encontre. Esse segundo caso é chamado de busca recursiva, isto é, o primeiro servidor DNS vai procurando noutros servidores até encontrar a resposta.

Outra grande funcionalidade de um servidor DNS numa rede é realizar melhoria na velocidade da ligação com a internet, fazendo *cache* de DNS. Funciona da seguinte forma: cada vez que um utilizador acede a um site, o servidor DNS verifica se o site acedido está presente numa lista. Caso esteja, o servidor vai armazenar uma série de informações desse *site* (como a estrutura do *site*, por exemplo, que é uma informação que não muda com muita frequência) e, a partir de então, caso outro utilizador aceda àquele site, essas informações serão carregadas do servidor DNS, diminuindo a quantidade de informações que serão acedidas via internet. Existem informações, entretanto, que não permitem o armazenamento através da *cache*. Por exemplo, um site de notícias deve ser carregado completamente pela *Internet*, procurando no servidor onde o site está, todas as suas informações. Ele pode trazer informações como notícias atualizadas que mudam com grande frequência. Porém, a estrutura do *site*, ou seja, a disposição das informações no ecrã raramente muda e, isso sim, pode ser armazenado na cache do servidor DNS.

Ainda assim, nesse caso, essas informações são poucas e não imprimem grande diferença na velocidade da ligação, dependendo da quantidade dos utilizadores da rede. Informações como vídeos que normalmente consomem grande quantidade de banda, ao serem armazenados em uma cache no servidor DNS, fazem com que, um vídeo seja acedido pela primeira vez e armazenado na *cache*, se acedido posteriormente, será carregado do servidor DNS, pois o vídeo é o mesmo, não havendo necessidade de carregá-lo da Internet novamente.

O servidor DNS também opera quando um e-mail é enviado ou uma transação FTP (*File Transfer Protocol*) é executada.

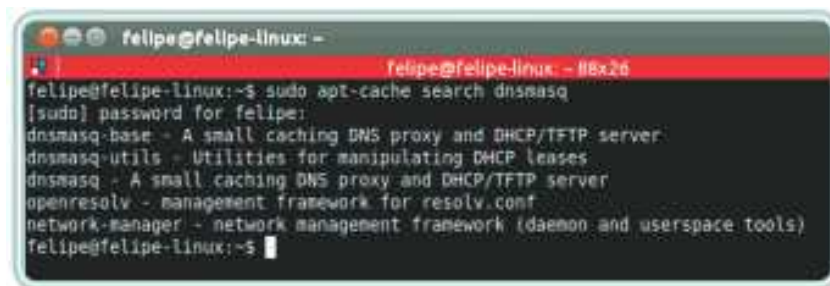
Instalação

A instalação de pacotes realizada através do gestor de pacotes é mais prática e fácil, pois, dessa forma, as dependências necessárias para os aplicativos que estão a ser instalados, são instaladas automaticamente, trazendo melhor compatibilidade entre as versões dos pacotes possíveis.

É importante lembrar que os exemplos são baseados no sistema operativo Ubuntu, que é uma distribuição Linux.

Apesar de existirem várias aplicações capazes de fazer o trabalho de um servidor DNS, será abordado apenas o **dnsmasq**, um dos mais conhecidos e utilizados atualmente.

Para a instalação, basta pesquisar pelos respetivos pacotes das aplicações no repositório de pacotes do Ubuntu, através do comando `apt-cache`. A Figura 6.1 mostra o resultado da busca por pacotes do **dnsmasq**.



```
felipe@felipe-linux: ~  
felipe@felipe-linux: ~$ sudo apt-cache search dnsmasq  
[sudo] password for felipe:  
dnsmasq-base - A small caching DNS proxy and DHCP/TFTP server  
dnsmasq-utils - Utilities for manipulating DHCP leases  
dnsmasq - A small caching DNS proxy and DHCP/TFTP server  
openresolv - management framework for resolv.conf  
network-manager - network management framework (daemon and userspace tools)  
felipe@felipe-linux: ~$
```

Figura 52-Pesquisa por pacotes do **dnsmasq**

Configuração

A princípio, o **dnsmasq** funciona ao ser instalado e iniciado, caso já exista um servidor DNS na rede, pois a sua configuração padrão permite o seu funcionamento. Entretanto, pode ser necessário alterar a configuração para atender a alguma necessidade de rede. Para isso, basta a edição do arquivo **/etc/dnsmasq.conf**. A Figura 53 mostra algumas modificações que podem ser realizadas para personalizar o **dnsmasq**. O manual do **dnsmasq** traz mais alterações.

Parâmetro	Descrição
domain-needed	Esse parâmetro evita que requisições sem domínio sejam enviadas. Assim, é possível, por exemplo, acessar recursos da rede local, através do endereço IP de uma máquina da rede local.
filterwin2k	Realiza filtros a requisições que computadores com sistema Microsoft Windows 2000/XP geram automaticamente.
resolve-else	Por padrão, os endereços de IP dos outros servidores DNS que serão consultados, ficam no arquivo /etc/resolv.conf. Porém, através deste parâmetro é possível indicar outro endereço.
address	Permite definir um endereço IP fixo para um determinado endereço de site. Ativando essa linha, sempre que o site configurado for acessado, o endereço IP configurado vai ser acessado. Assim é possível, por exemplo, bloquear sites, colocando um endereço IP de um servidor web que mostre uma página, informando que aquele site está bloqueado.
listen-address	É o endereço IP da interface na qual dnsmasq vai monitorar. Como a configuração é realizada no próprio servidor, o endereço colocado deve ser o da interface loopback (127.0.0.1).
port	Porta do servidor DNS (a porta padrão é a 53).
dns-forwards-max	Número máximo de requisições DNS concorrentes que o servidor irá responder (padrão é 150).

Figura 53-Alguns parâmetros de configuração do dnsmasq

Na Figura 53, mencionou-se que os servidores DNS a serem consultados são inseridos, por padrão, no arquivo **/etc/resolv.conf**. Para inserir um servidor DNS no arquivo, basta inserir uma linha com a estrutura **“nameserver endereço-ip”**. Portanto, é importante colocar uma linha que acrescente o endereço do próprio servidor para que ele seja consultado, inserindo uma linha contendo **“nameserver 127.0.0.1”**.

Após uma modificação no arquivo de configuração é necessário reiniciar o serviço através do comando **sudo /etc/init.d/dnsmasq restart** ou **sudo service dnsmasq restart**.

Todo sistema operativo Linux possui uma interface de rede virtual chamada loopback (lo) que serve para indicar o próprio computador. Portanto, para aceder a um recurso instalado no próprio computador, usa-se essa interface, cujo endereço IP é 127.0.0.1.

Cache de DNS

Uma das grandes aplicações do servidor DNS é a economia de banda. Quando um utilizador acede a uma página de *Internet*, ele recebe informações diversas sobre essa página. Dentre essas informações, está a estrutura da página, ou seja, o posicionamento das informações no ecrã, da mesma forma que as cores padrões da página. Junto com isso, vêm as informações sobre o conteúdo da página que é atualizado constantemente. No caso de uma página de notícias, a estrutura seria como a notícia fica disposta no ecrã e vai ser igual para todas as notícias dia após dia. O conteúdo seria a notícia em si. Essa informação muda sempre que uma nova notícia é inserida no site. Dessa forma, sem um servidor de cache de DNS, sempre que um utilizador aceder a um *site*, todas as informações estruturais e de conteúdo serão acedidos diretamente do servidor no qual o *site* está hospedado. O uso de um servidor *cache* de DNS faz com que, quando qualquer utilizador da rede aceder a uma página, as informações sobre a estrutura dessa página fiquem armazenadas no servidor. Assim, na próxima vez que um utilizador aceder a essa página, parte das informações serão carregadas do servidor de *cache* de DNS, economizando banda. Já as informações sobre o conteúdo da página, por mudarem com certa frequência, são carregadas diretamente do servidor do site. Essa solução, numa rede grande, com um número elevado de utilizadores é muito útil, pois utilizadores aceder à mesma página várias vezes ao dia provocam grande consumo de banda. Outro uso comum do servidor de cache de DNS é para armazenamento de arquivos e vídeos. Quando um utilizador acede um determinado vídeo num *site*, o servidor o armazena e quando outro utilizador aceder o mesmo vídeo, este já estará guardado no servidor. Outro exemplo são atualizações de sistema, onde vários arquivos são descarregados de um

determinado servidor para atualizar aplicações e sistemas operativos. Esses arquivos podem ser armazenados no cache do DNS e, quando solicitados novamente, são carregados.

Para configurar um servidor de *cache* de DNS, inicialmente é necessário inserir uma linha no arquivo **/etc/resolv.conf**. O arquivo **/etc/resolv.conf** contém os endereços dos servidores DNS que o computador irá utilizar. É necessário, portanto, inserir, nesse arquivo, antes de todos os outros servidores, o endereço do próprio servidor (assumindo que essa configuração está a ser realizada no próprio servidor), uma linha contendo o conteúdo “nameserver 127.0.0.1”. É necessário também, inserir uma linha contendo “listen-address = 127.0.0.1” no arquivo **/etc/dnsmasq.conf**. As informações dessas linhas indicam que quando uma página de internet for solicitada, o servidor de DNS verificará o endereço 127.0.0.1, nele próprio, ou seja, onde as informações do cache estão armazenadas.

Para testar o servidor, basta utilizar o comando “**dig endereço-do-site**”. Esse comando vai mostrar o tempo em milissegundos que foi necessário para realizar a requisição do site e ter a sua resposta. Feito isso, basta realizar o comando novamente com o mesmo site, visto que na primeira execução, o cache de DNS armazenou parte das informações desse site e na segunda, essas informações foram carregadas do servidor de cache de DNS. A Figura 54 mostra a primeira execução do comando, enquanto a Figura 55 mostra a segunda execução.

```
root@felipe-linux:/home/felipe# dig www.yahoo.com.br

: <<> DiG 9.7.0-P1 <<> www.yahoo.com.br
;; global options: +cmd
;; Got answer:
;; ->HEADER<<- opcode: QUERY, status: NOERROR, id: 35768
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.yahoo.com.br.      IN      A

;; ANSWER SECTION:
www.yahoo.com.br.      1638    IN      CNAME   rc.yahoo.com.
rc.yahoo.com.          217     IN      CNAME   rc.g01.yahoodns.net.
rc.g01.yahoodns.net.   220     IN      CNAME   any-rc.a01.yahoodns.net.
any-rc.a01.yahoodns.net. 126     IN      A       98.139.102.145
any-rc.a01.yahoodns.net. 126     IN      A       68.180.206.184

;; Query time: 83 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Feb 15 22:09:56 2012
;; MSG SIZE rcvd: 150
```

Figura 54-Primeira execução do comando dig

```
root@felipe-linux: /home/felipe 80x20
root@felipe-linux:/home/felipe# dig www.yahoo.com.br

; <<>> DiG 9.7.0-P1 <<>> www.yahoo.com.br
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27631
;; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.yahoo.com.br.                IN      A

;; ANSWER SECTION:
www.yahoo.com.br. 57      IN      A      98.139.102.145

;; Query time: 3 msec
;; SERVER: 127.0.0.1(127.0.0.1)
;; WHEN: Wed Feb 15 22:11:04 2012
;; MSG SIZE rcvd: 50

root@felipe-linux:/home/felipe#
```

Figura 55-Segunda execução do comando dig

É possível observar nas Figuras 54 e 55 a diferença entre o tempo da consulta. A primeira vez em que o site foi acessado pelo comando dig, o tempo gasto foi 83 milissegundos; na segunda, o tempo foi de apenas 6.3 milissegundos.

Resumo

Apresentou-se o funcionamento de um servidor DNS, trazendo o seu conceito e aplicação. Para realizar uma abordagem prática, explanou-se o processo de instalação em um sistema GNU/Linux Ubuntu.

Em seguida, os parâmetros de configuração do servidor DNS foram apresentados, juntamente com um passo a passo para realizar as configurações. Por fim, foi explicada uma das aplicações de um servidor DNS, A *cache* de DNS, que tem como objetivo economizar banda da rede. A *cache* de DNS é muito útil em redes com uma grande quantidade de utilizadores. Por fim, foi mostrado o servidor DNS em funcionamento, quando se demonstrou que através do comando **dig** é visível a economia de banda em apenas um pedido.

Servidores de correio eletrônico

Objetivos

Instalar e configurar um servidor de correio eletrônico e dominar conhecimentos sobre o seu funcionamento.

Considerações iniciais

Servidores de correio eletrônico ou simplesmente servidores de e-mail (*Electronic Mail*) têm como objetivo oferecer uma solução de correio eletrônico corporativo personalizado, possibilitando personalizar o domínio do servidor. Outro uso muito comum para servidores de correio eletrônico é a eliminação de acesso a recursos externos, aumentando a produtividade, pois os utilizadores não têm acesso ao correio eletrônico pessoal. Numa rede com um grande número de utilizadores, através de uma boa gestão, o uso de um servidor de correio eletrônico economiza banda da ligação com a *Internet*, pois os utilizadores irão aceder aos recursos do correio eletrônico em execução num servidor dentro da rede e não em um servidor na *Internet*.

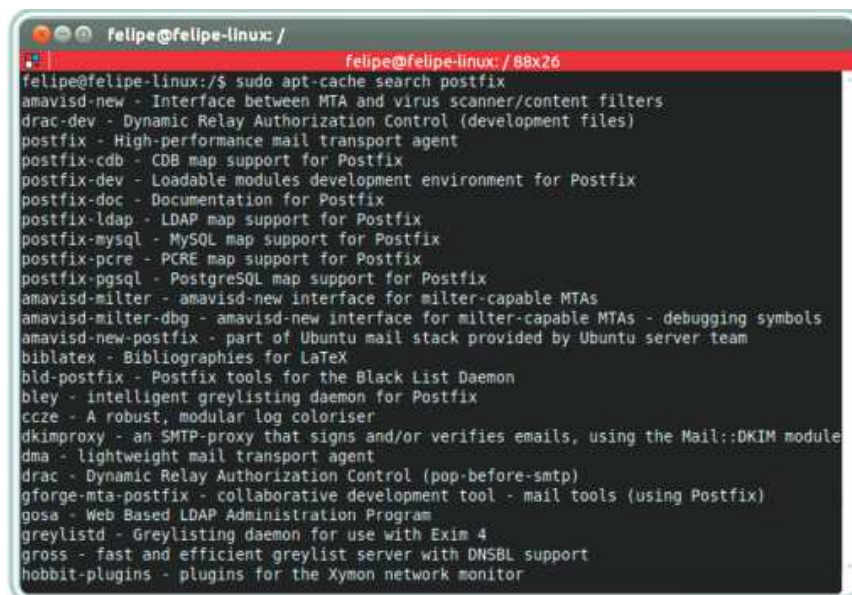
Em sistemas Linux, existem vários servidores de correio eletrônico que, na maioria das vezes, tornam-se bastante complexos na sua configuração. Um dos mais utilizados e que se irá abordar é o **Postfix**.

Instalação

Conforme se abordou anteriormente, a instalação de pacotes realizada através de gestores de pacotes é mais prática e fácil, pois dessa forma, as dependências necessárias para as aplicações que estão a ser instalados são instaladas automaticamente, trazendo compatibilidade entre as versões dos pacotes possíveis.

É importante lembrar que os exemplos são baseados no sistema operativo Ubuntu, que é uma distribuição Linux.

A instalação do Postfix dá-se da mesma maneira abordada anteriormente. A Figura 56 mostra os pacotes disponíveis no repositório, através do comando apt-cache.



```
felipe@felipe-linux: /  
felipe@felipe-linux: / 88x26  
felipe@felipe-linux:/$ sudo apt-cache search postfix  
amavisd-new - Interface between MTA and virus scanner/content filters  
drac-dev - Dynamic Relay Authorization Control (development files)  
postfix - High-performance mail transport agent  
postfix-cdb - CDB map support for Postfix  
postfix-dev - Loadable modules development environment for Postfix  
postfix-doc - Documentation for Postfix  
postfix-ldap - LDAP map support for Postfix  
postfix-mysql - MySQL map support for Postfix  
postfix-pcre - PCRE map support for Postfix  
postfix-pgsql - PostgreSQL map support for Postfix  
amavisd-milter - amavisd-new interface for milter-capable MTAs  
amavisd-milter-dbg - amavisd-new interface for milter-capable MTAs - debugging symbols  
amavisd-new-postfix - part of Ubuntu mail stack provided by Ubuntu server team  
bibtex - Bibliographies for LaTeX  
bld-postfix - Postfix tools for the Black List Daemon  
bley - intelligent greylisting daemon for Postfix  
ccze - A robust, modular log coloriser  
dkimproxy - an SMTP-proxy that signs and/or verifies emails, using the Mail::DKIM module  
dma - lightweight mail transport agent  
drac - Dynamic Relay Authorization Control (pop-before-smtp)  
gforge-mta-postfix - collaborative development tool - mail tools (using Postfix)  
gosa - Web Based LDAP Administration Program  
greylistd - Greylisting daemon for use with Exim 4  
gross - fast and efficient greylist server with DNSBL support  
hobbit-plugins - plugins for the Xymon network monitor
```

Figura 56-Pesquisa de pacotes do Postfix no repositório

É possível depreender pelas que a instalação das aplicações sé extremamente simples, bastando um comando no terminal do Linux. É importante lembrar que a instalação através do gestor de pacotes traz os pacotes mais estáveis e designados para aquela distribuição. Por esse motivo, a versão dos pacotes instalados pode não ser a mais atual e, caso seja necessária alguma funcionalidade presente apenas em versão mais atual, uma instalação manual é necessária. Ainda assim, na maioria dos casos, os pacotes presentes nos repositórios são suficientes para satisfazer as necessidades dos utilizadores.

Durante a instalação, o Postfix faz duas perguntas para o utilizador. A primeira, presente na Figura 57, é sobre a função do servidor, se ele apenas receberá mensagens ou se também as enviará. No caso de ele apenas receber, obviamente deve haver outro servidor que realize a função de envio. Por isso, com o objetivo de evitar a instalação de outro serviço ou servidor, a opção “Internet site” (geralmente mais usada) faz todo o trabalho, tanto o de enviar, quanto o de receber mensagens.

Já a segunda pergunta (Figura 58) é sobre o domínio que será utilizado pelas mensagens enviadas pelo servidor. Caso exista um domínio já registado, basta inseri-lo. Caso não tenha um domínio registado, basta que se defina um.

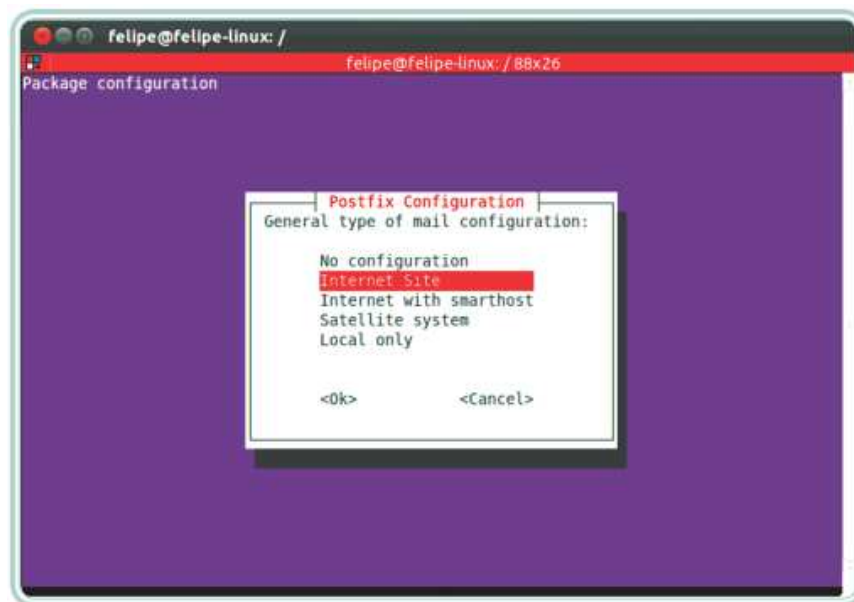


Figura 57-Função do servidor

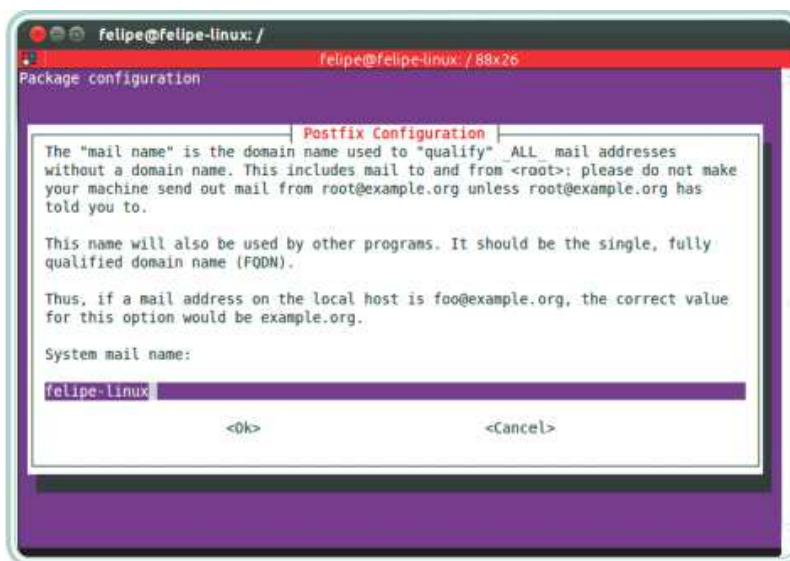


Figura 58-Domínio do servidor

Configuração

Após a instalação, é necessária a configuração do Postfix, que se dá através da edição de arquivos de configuração, do mesmo modo que as aplicações apresentadas anteriormente. Os arquivos de configuração do Postfix ficam na pasta **/etc/postfix/**. Dentro dela existe o arquivo **main.cf** que é o arquivo de configuração. Para editá-lo, basta usar um editor de texto qualquer, como o Nano, por exemplo, através do comando **sudo nano /etc/postfix/main.cf**.

A Figura 59 mostra os parâmetros de configuração e sua descrição.

Parâmetro	Descrição
myhostname	Nome do host no domínio. Ex.: meuhost.meudominio.com
mydomain	Nome do domínio. Ex.: meudominio.com
myorigin	Deve-se comentar o parâmetro myhostname inserindo o símbolo # no início da linha e deixar mydomain descomentado. Ex.: myorigin=\$mydomain. Isso significa que myorigin terá o mesmo valor que mydomain.
inet_interfaces	Define quem terá acesso ao servidor. Para permitir acesso a todos os usuários da rede, esse parâmetro deve ser definido como "all". Caso seja necessário definir os computadores que terão acesso, basta inserir os endereços IPs, separados por vírgula.
mydestination	Define os destinos com os quais o servidor pode comunicar-se. Uma forma de definir que esse servidor poderá enviar mensagens livremente dentro do domínio, é definindo-o. \$myhostname, localhost.\$mydomain, \$mydomain, mail.\$mydomain, www.\$mydomain, ftp.\$mydomainmydestination".
relay_domains	Determina os domínios para retransmissão. É bom definir como \$mydomain.

Figura 59-Parâmetros de configuração

Conforme se mencionou anteriormente, a configuração de um servidor de correio eletrônico pode tornar-se algo extremamente complexo, dependendo de cada caso. Para mais detalhes sobre os parâmetros de configuração basta consultar o manual do Postfix.

Após o término da configuração, é necessário reiniciar o serviço do Postfix. No caso deste servidor, o processo é um pouco diferente dos que se mostrou anteriormente, necessitando dos seguintes comandos:

- a) `sudo service postfix stop` (ou `sudo /etc/init.d/postfix stop`);
- b) `sudo service postfix start` (ou `sudo /etc/init.d/postfix start`);
- c) `sudo service postfix reload` (ou `sudo /etc/init.d/postfix reload`).

Para permitir a comunicação entre o servidor e os clientes, ou seja, para permitir que um cliente receba as mensagens eletrônicas é necessário executar o comando **`sudo /etc/init.d/inet start`** ou **`sudo service inet start`** para iniciar o serviço de rede do servidor.

Dentro do diretório **`/var/spool`** fica a fila de mensagens que será manipulada pelo Postfix. Como esse é um diretório do sistema, é necessário alterar as suas permissões para que o utilizador Postfix (criado na instalação) possa alterá-lo. Para isso, é utilizado o comando `chown`, da seguinte forma: **`sudo chown <utilizador-que-terá-permissões> -R /var/spool/postfix`**. O diretório **`/var/spool/postfix`** no final do comando é o diretório no qual as permissões serão aplicadas. Já o parâmetro `-R` significa que as permissões serão aplicadas recursivamente, ou seja, todas as pastas dentro da pasta `/var/spool/postfix` receberão a nova permissão. No caso, o comando ficará: **`sudo chown postfix -R /var/spool/postfix`**.

Para que o utilizador root do sistema Linux receba mensagens com alertas do sistema, basta criar um arquivo chamado “aliasas” na pasta **`/etc`**, com o comando **`sudo nano /etc/aliasas`**. Dentro dele, basta inserir uma linha com o seguinte conteúdo: **`“root: administrador@dominio”`**, onde administrador é o nome do utilizador que receberá as mensagens, e domínio é o domínio configurado na instalação do Postfix.

Depois basta executar o comando **`sudo newaliasas`** para recarregar as informações inseridas no arquivo.

Por fim, basta inserir os utilizadores através do comando **`adduser`** do Linux.

É possível também, inserir os utilizadores numa base de dados do MySQL.

Resumo

Trouxe-se a apresentação de um servidor de correio eletrônico. Servidores de correio eletrônico são bastante utilizados em empresas para direcionar ao uso de ferramentas profissionais dos seus funcionários, pois a utilização de um correio eletrônico da empresa faz com que os utilizadores não precisem utilizar um sistema de correio eletrônico pessoal, misturando assuntos pessoais e profissionais.

Abordaram-se a instalação e a configuração de um servidor de correio eletrônico, explicando os parâmetros de configuração do serviço.

Vale ressaltar que o servidor de correio eletrônico Postfix possui uma ótima integração com outras ferramentas de rede, tornando-o assim, mais útil. Entretanto, essas integrações aumentam a complexidade de sua instalação, configuração e gestão.

Servidor de partilha de arquivos e servidor de impressão

Objetivos

Instalar e configurar um servidor de partilha de arquivos Samba, da mesma forma, conhecer sobre o seu funcionamento e finalidade

Considerações iniciais

A partilha de arquivos entre computadores é muito comum em redes corporativas e empresariais, principalmente em casos onde um mesmo arquivo é utilizado por uma grande parte dos utilizadores da rede. Um exemplo comum é a partilha de instaladores de programas. Isso faz-se mantendo uma pasta partilhada num servidor na qual os instaladores de todas as aplicações utilizados pelos utilizadores da rede ficam armazenados. Assim, quando um utilizador precisar de um desses instaladores, basta aceder à pasta e copiá-lo. A utilização de perfis de rede também é muito usada. Isso funciona criando um servidor gestor de domínio, colocando todos os computadores da rede nesse domínio e fazendo os utilizadores acederem os computadores através de uma autenticação de utilizador e palavra-passe. Em outras palavras, os utilizadores irão inserir nos computadores, um utilizador e palavra-passe que estarão armazenados no servidor e assim poderão aceder a qualquer computador da rede que esteja no domínio. Então, várias informações do utilizador serão carregadas do servidor e, independentemente do computador que ele utilizar, as suas configurações, históricos de navegação e outras informações pessoais serão carregadas no computador acedido. Ainda é possível, por parte dos administradores da rede, realizarem uma série de controlo dos computadores da rede. Por exemplo, um utilizador autenticado num computador através de um domínio, permite que o utilizador utilize qualquer aplicação normalmente, porém impede (desde que configurado adequadamente) que ele instale ou desinstale uma aplicação ou realize operações em nível de administrador, evitando assim, que aplicações desnecessárias sejam instaladas nos computadores, ocupando espaço.

No sistema operativo Linux, o servidor responsável pela partilha de arquivos mais utilizado é o **Samba**. Ele permite a partilha de arquivos entre computadores com sistemas operativos GNU/Linux e Microsoft Windows.

Um dos grandes atrativos do Samba é a compatibilidade com uma gama de outras aplicações, o que permite a adição de várias funcionalidades do servidor. Por exemplo, o Samba permite que outros tipos de autenticação sejam utilizados, pois pode acontecer de ele ser implementado numa rede já existente, com um servidor de autenticação já operacional. A utilização de um novo sistema de autenticação obrigará os administradores da rede a refazerem o registo de todos os utilizadores, o que seria extremamente trabalhoso e, no caso de uma grande rede, completamente inviável. O Samba permite ainda que cada utilizador tenha uma pasta no servidor que, ao autenticar-se num computador da rede, passe a ser acessível, como se fosse uma pasta local do computador, que permite ao utilizador guardar nela os seus arquivos pessoais. Essa pasta ainda pode ser acessível pela rede, mediante autenticação de utilizador e palavra-passe. Ainda é possível definir uma cota para cada utilizador, de modo que a pasta terá um espaço disponível predeterminado pelo administrador da rede. E, devido à existência de vários níveis de utilizadores dentro de uma rede, utilizadores de diferentes níveis podem ter diferentes cotas de espaço em disco no servidor. Por exemplo, a implementação de um servidor Samba numa rede corporativa, na qual existe um utilizador, presidente da empresa e outro funcionário, permite ao presidente ter uma cota de espaço em disco maior que a cota do funcionário.

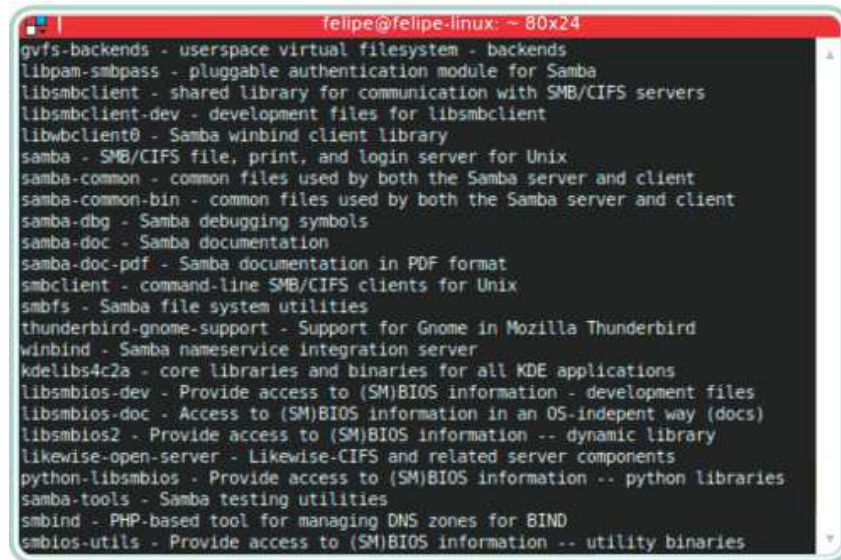
O Samba, ou outro servidor que tenham o mesmo propósito numa rede corporativa, é muito comum, pois eles oferecem várias maneiras de gerir a rede e controlar o acesso dos utilizadores. Numa rede com um servidor **Microsoft Windows**, o servidor que tem esse propósito chama-se **Controlador de**

Domínio. Já em redes onde o servidor que tem essa função é um servidor **Linux**, ele é denominado **PDC (Personal Domain Controller)** ou **Controlador de Domínio Pessoal**.

Instalação

Como foi abordado anteriormente, a instalação de pacotes realizada através de um gestor de pacotes é mais prática e fácil, pois, dessa forma, as dependências necessárias para as aplicações são instaladas automaticamente, trazendo assim melhor compatibilidade entre as versões dos pacotes possíveis.

A instalação do Samba ocorre da mesma forma que a abordada anteriormente. A Figura 60 mostra os pacotes disponíveis no repositório, através do comando **apt-cache search samba**.



```
felipe@felipe-linux: ~ 80x24
gvfs-backends - userspace virtual filesystem - backends
libpam-smbpass - pluggable authentication module for Samba
libsmbclient - shared library for communication with SMB/CIFS servers
libsmbclient-dev - development files for libsmbclient
libwbclient0 - Samba winbind client library
samba - SMB/CIFS file, print, and login server for Unix
samba-common - common files used by both the Samba server and client
samba-common-bin - common files used by both the Samba server and client
samba-dbg - Samba debugging symbols
samba-doc - Samba documentation
samba-doc-pdf - Samba documentation in PDF format
smbclient - command-line SMB/CIFS clients for Unix
smbfs - Samba file system utilities
thunderbird-gnome-support - Support for Gnome in Mozilla Thunderbird
winbind - Samba nameservice integration server
kdelibs4c2a - core libraries and binaries for all KDE applications
libsmbios-dev - Provide access to (SM)BIOS information - development files
libsmbios-doc - Access to (SM)BIOS information in an OS-independent way (docs)
libsmbios2 - Provide access to (SM)BIOS information -- dynamic library
likewise-open-server - Likewise-CIFS and related server components
python-libsmbios - Provide access to (SM)BIOS information -- python libraries
samba-tools - Samba testing utilities
smbind - PHP-based tool for managing DNS zones for BIND
smbios-utils - Provide access to (SM)BIOS information -- utility binaries
```

Figura 60-Pesquisa de pacotes do Samba no repositório

Após verificar os nomes dos pacotes, basta instalá-los através do comando **sudo apt-get install <nome-do-pacote>**.

Configuração

A configuração do Samba consiste na alteração de parâmetros presentes no arquivo **/etc/samba/smb.conf**. Para editá-lo, basta utilizar um editor de texto qualquer, como por exemplo, o Nano, através do comando **sudo nano /etc/samba/smb.conf**.

O arquivo de configuração do Samba possui uma estrutura especial para determinar o funcionamento de várias funcionalidades do Samba. O arquivo deve ser dividido em seções. Para indicar o início de uma seção, utiliza-se o nome da seção entre parêntesis retos. Ex.: [global]

Após a determinação do início de uma seção, todas as configurações referentes a essa seção devem ser declaradas, através dos parâmetros. Ao término das configurações de todos os parâmetros da seção, inicia-se uma nova seção, declarando seu nome entre parêntesis. A Figura 61 exemplifica a estrutura do arquivo de configuração do Samba.


```

1. [seção1]
2. parâmetro1
3. parâmetro2
4. .
5. .
6. .
7. parâmetroN
8.
9. [seção2]
10. parâmetro1
11. parâmetro2
12. .
13. .
14. .
15.
16. [seçãoN]
17. parâmetro1
18. parâmetro2
19. .
20. .
21. .
22. parâmetroN

```

Figura 61-Estrutura do arquivo de configuração do Samba

A primeira seção discutida é a seção global que determina configurações do servidor como segurança, forma de acesso, nome, grupo de trabalho, entre outras. Para iniciá-la, basta declará-la com **[global]** logo no início do arquivo. A Figura 62 mostra os parâmetros mais utilizados para configurar essa seção, juntamente com suas descrições.

Parâmetro	Descrição
workgroup	Grupo de trabalho.
server string	Nome do servidor na rede.
security	Método de autenticação para o acesso. Por exemplo, configurando como <i>user</i> determina que o acesso seja através de usuário e senha e configurando como <i>share</i> , o acesso será sem usuário e senha.
comment	Comentário para o servidor.

Figura 62-Parâmetros da seção [global] do arquivo de configuração do Samba

Serão descritas mais três seções bastante utilizadas, pela sua grande utilidade. É importante que por meio da explicação delas, que contêm configurações de permissão de pastas, é possível criar outras seções, alterando informações julgadas necessárias.

Essas três seções são correspondentes à pasta pessoal dos utilizadores, uma pasta pública, que seja acessível a qualquer um, mediante utilizador e palavra-passe. Nesse caso, qualquer utilizador que aceder à pasta verá o mesmo conteúdo, mas não terá permissão para alterá-lo. A terceira seção é referente às impressoras.

Para criar uma seção para uma pasta pública, basta iniciar uma seção com um nome qualquer, usado para descrever o conteúdo da pasta. Neste exemplo, a pasta será destinada a guardar instaladores de programas e, por isso, o seu nome será “instaladores”. Então, para iniciar a nova seção, basta inserir, após o último parâmetro da seção [global], uma linha contendo **[instaladores]**.

A Figura 63 mostra os parâmetros mais utilizados e sua descrição, para configurar a seção do exemplo de instaladores.

Parâmetro	Descrição
comment	Comentário da pasta. Ex.: Instaladores.
path	Diretório da pasta no servidor. Ex.: /home/samba/Instaladores.
public	Se esta for uma pasta visível a outros usuários da rede.
browseable	Se a pasta ficará visível na rede.
writable	Se a pasta permitirá escrita.
read only	Se a pasta permitirá apenas leitura.
valid users	Usuários que poderão acessar a pasta.

Figura 63-Parâmetros de configuração da secção de instaladores

É importante lembrar que a pasta descrita no parâmetro **path** deve ser criada, caso não exista, pois, o Samba não a cria automaticamente.

A segunda seção, responsável pelo acesso à pasta pessoal de cada utilizador, inicia-se com a palavra “**homes**” entre parêntesis retos. Assim, o Samba interpreta que aquela seção determinará a configuração do acesso às pastas dos utilizadores. Ela deve vir após o último parâmetro de uma seção qualquer, seguindo a estrutura apresentada na Figura 63.

A Figura 64 mostra os parâmetros e suas descrições na seção [homes].

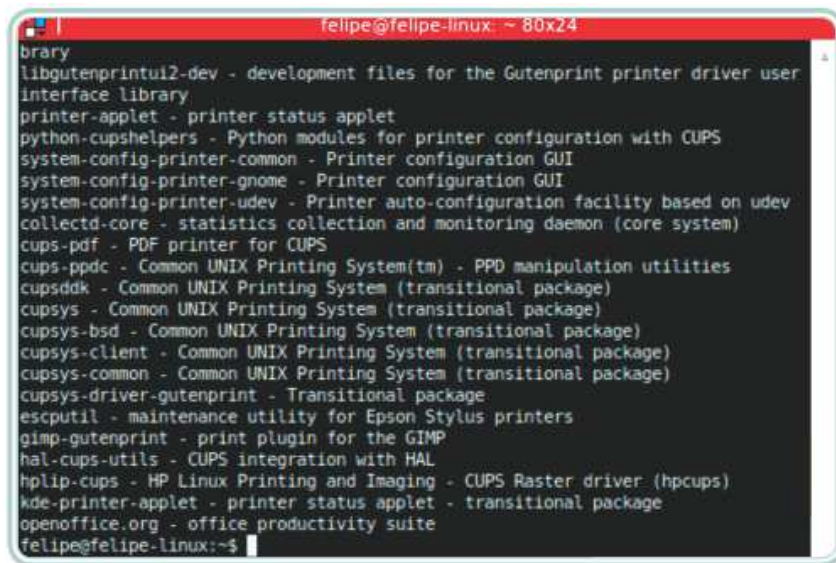
Parâmetro	Descrição
comment	Comentário da pasta.
public	Se a pasta será visível para outros usuários. Nesse caso, esse parâmetro pode ser definido como “no”. Assim, somente o usuário dono da pasta poderá ver a mesma.
browseable	Se a pasta ficará visível na rede.
writable	Se a pasta permitirá escrita.

Figura 64-Parâmetros de configuração da secção referentes às pastas dos utilizadores

As pastas dos utilizadores são criadas quando o utilizador é criado. As pastas dos utilizadores ficam no diretório /home/<nome-do-utilizador> e, por isso, não é necessário informar o caminho, conforme foi feito na seção de Instaladores. Além do mais, para cada utilizador, a pasta muda. Por exemplo, quando o utilizador “pedro” aceder à sua pasta pessoal, o diretório carregado pelo Samba será /home/pedro. Já quando o utilizador “paulo” aceder à sua pasta pessoal, a pasta carregada será /home/paulo.

A terceira e última seção é relacionada à impressão e define como partilhar uma impressora na rede. Antes de realizar a partilha, deve-se instalar a impressora no servidor, para depois poder partilhá-la. Em sistemas operativos Linux, o responsável por gerir as impressoras é o **cups** que, ao ser instalado, oferece uma maneira de gerir impressoras e impressões.

A Figura 65 mostra os pacotes do cups disponíveis no repositório do Ubuntu.

A terminal window titled 'felipe@felipe-linux: ~ 80x24' displays a list of CUPS-related packages. The list includes: 'brary', 'libgutenprintui2-dev' (development files for the Gutenprint printer driver user interface library), 'printer-applet' (printer status applet), 'python-cupshelpers' (Python modules for printer configuration with CUPS), 'system-config-printer-common' (Printer configuration GUI), 'system-config-printer-gnome' (Printer configuration GUI), 'system-config-printer-udev' (Printer auto-configuration facility based on udev), 'collectd-core' (statistics collection and monitoring daemon (core system)), 'cups-pdf' (PDF printer for CUPS), 'cups-ppdc' (Common UNIX Printing System(tm) - PPD manipulation utilities), 'cupsddk' (Common UNIX Printing System (transitional package)), 'cupsys' (Common UNIX Printing System (transitional package)), 'cupsys-bsd' (Common UNIX Printing System (transitional package)), 'cupsys-client' (Common UNIX Printing System (transitional package)), 'cupsys-common' (Common UNIX Printing System (transitional package)), 'cupsys-driver-gutenprint' (Transitional package), 'escputil' (maintenance utility for Epson Stylus printers), 'gimp-gutenprint' (print plugin for the GIMP), 'hal-cups-utils' (CUPS integration with HAL), 'hplip-cups' (HP Linux Printing and Imaging - CUPS Raster driver (hpcups)), 'kde-printer-applet' (printer status applet - transitional package), and 'openoffice.org' (office productivity suite). The prompt 'felipe@felipe-linux:~\$' is visible at the bottom.

```
felipe@felipe-linux: ~ 80x24
brary
libgutenprintui2-dev - development files for the Gutenprint printer driver user
interface library
printer-applet - printer status applet
python-cupshelpers - Python modules for printer configuration with CUPS
system-config-printer-common - Printer configuration GUI
system-config-printer-gnome - Printer configuration GUI
system-config-printer-udev - Printer auto-configuration facility based on udev
collectd-core - statistics collection and monitoring daemon (core system)
cups-pdf - PDF printer for CUPS
cups-ppdc - Common UNIX Printing System(tm) - PPD manipulation utilities
cupsddk - Common UNIX Printing System (transitional package)
cupsys - Common UNIX Printing System (transitional package)
cupsys-bsd - Common UNIX Printing System (transitional package)
cupsys-client - Common UNIX Printing System (transitional package)
cupsys-common - Common UNIX Printing System (transitional package)
cupsys-driver-gutenprint - Transitional package
escputil - maintenance utility for Epson Stylus printers
gimp-gutenprint - print plugin for the GIMP
hal-cups-utils - CUPS integration with HAL
hplip-cups - HP Linux Printing and Imaging - CUPS Raster driver (hpcups)
kde-printer-applet - printer status applet - transitional package
openoffice.org - office productivity suite
felipe@felipe-linux:~$
```

Figura 65-Pacotes do cups presentes no repositório

Após a instalação é necessário configurar o cups, alterando o arquivo **/etc/cups/cupsd.conf**. Nesse arquivo, a secção **“admin”** determina, além de outras coisas, quem tem permissão de acesso às impressoras. Portanto, para permitir que os utilizadores possam gerir as impressoras instaladas no servidor, essa secção deve ser alterada. A alteração consiste basicamente em inserir uma linha na qual se habilita a permissão dos computadores da rede (ou de apenas um computador) a acederem e gerirem as impressoras. Essa linha deve conter a estrutura **“Allow <endereço-ip>”**, onde endereço-ip é o endereço IP do(s) computador(es) que poderá(ão) aceder às impressoras. Por exemplo, para inserir o computador com endereço IP 192.168.10.10, basta inserir a linha **“Allow 192.168.10.10”**.

É importante lembrar que após as alterações é necessário reiniciar o serviço de impressão através do comando **“sudo /etc/init.d/cups restart”** ou **sudo service cups restart**.

O cups também oferece uma interface acessível via navegador para gestão de impressoras, visto que, muitas vezes, o servidor não possui interface gráfica. Assim, instalar e gerir impressoras fica mais fácil. Para aceder ao gestor de impressoras do cups, basta escrever no navegador o endereço IP do servidor, com a porta 631. Por exemplo, caso o endereço IP do servidor seja 192.168.10.1, basta escrever o endereço **http://192.168.10.1:631**. A Figura 66 mostra a interface de gestão do cups.

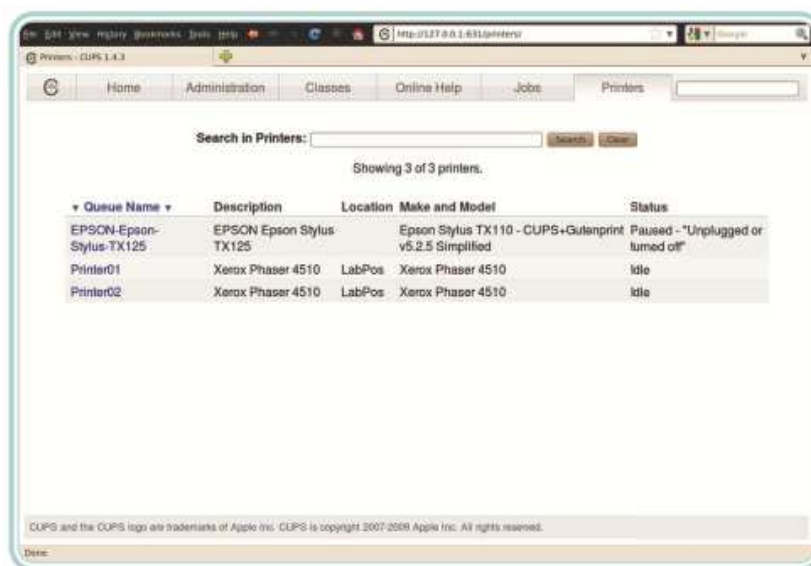


Figura 66-Interface de gestão de impressoras do cups

Com as impressoras devidamente instaladas no servidor, basta inserir duas linhas na seção **[global]** do Samba para que ele partilhe as impressoras. A Figura 67 mostra os parâmetros e sua descrição referentes a essas duas linhas.

Parâmetro	Descrição
printing	Indica quem vai ser o responsável por gerenciar as impressoras. No caso é o "cups". Portanto essa linha deve conter "printing cups".
load printers	Indica se as impressoras do gerenciador devem ser carregadas pelo Samba. Para carregá-las, basta definir esse parâmetro com "yes". Portanto essa linha deve conter "load printers yes" para carregar as impressoras e "load printers no" para não carregá-las.

Figura 67-Inserção e partilha de impressoras no Samba

Após a configuração da seção **[global]** do Samba, basta inserir uma nova seção referente às impressoras, com uma linha contendo **[printers]**, e configurar seus parâmetros presentes na Figura 68.

Parâmetro	Descrição
comment	Comentário para as impressoras.
print ok	Esse parâmetro define se as impressoras estarão compartilhadas ou não. Definindo-o com "yes" ativa o compartilhamento e definindo-o com "no" desativa.
guest ok	Permite que usuários que não estejam cadastrados no Samba imprimam, caso definido com o valor "yes". Definindo com o valor "no", apenas usuários cadastrados no Samba terão permissão para imprimir.
path	Caminho das impressoras. Normalmente, elas ficam em /var/spool/samba.
browseable	Define se as impressoras serão acessíveis caso os usuários acessem o servidor de impressão pela rede. Por exemplo, como o servidor tratado nesta aula possui uma pasta compartilhada (Instaladores), os usuários podem acessar esse computador pela rede e acessar assim essa pasta. Definindo esse parâmetro com "yes", o servidor também mostrará que possui as impressoras instaladas. Definindo com "no", não mostrará. Mesmo definindo esse parâmetro como "no", os usuários ainda poderão utilizar as impressoras normalmente.

Figura 68-Secção [Printers]

É possível também especificar esses mesmos parâmetros para cada impressora instalada no servidor. Pode haver caso onde apenas alguns utilizadores podem imprimir apenas numa impressora. Para isso, basta trocar a secção **[printers]** pela que contenha o nome da impressora. Assim, inserindo o parâmetro “**valid users**”, pode-se definir os utilizadores que utilizarão a impressora separando os seus nomes por vírgula. Por exemplo, a linha “**valid users = pedro, paulo**”, indica que apenas os utilizadores pedro e paulo podem utilizar as impressoras. Ainda é possível indicar os computadores que podem utilizar as impressoras através do parâmetro “**hosts allow**”, que funciona de forma semelhante ao “**valid users**”.

Esses parâmetros devem ser informados para todas as impressoras, inserindo uma nova secção no arquivo para cada uma.

Após a configuração de todas as secções do Samba, é necessário que o serviço seja reiniciado para que as alterações tenham efeito. A reinicialização do serviço pode ser realizada pelo comando **sudo restart smbd**. Por fim, basta registar os utilizadores no Samba. Caso o utilizador ainda não esteja registado no sistema, basta inseri-lo normalmente através do comando **adduser** e, depois de criá-lo ou se o mesmo já estiver presente no sistema, basta registá-lo no Samba com o comando **sudo smbpasswd <nome-do-utilizador>**. Isso finaliza a configuração do servidor Samba.

Posteriormente, basta inserir os outros computadores no domínio definido no servidor Samba, para que os utilizadores possam aceder às suas pastas armazenadas no servidor.

Resumo

Neste capítulo foram apresentadas instalações e configurações de um servidor de partilha de arquivos e de impressoras e a configuração dos serviços para situações comuns em redes corporativas.

O servidor de partilha de arquivos é bastante utilizado pela administração da rede, pois através dele é possível aceder a pastas compartilhadas que contenham aplicações e documentos necessários para realizar manutenção na rede.

O servidor de partilha de arquivos em sistemas GNU/Linux também é responsável pela partilha de impressoras. É importante deixar claro que o servidor de impressão é a ferramenta cups. É ele que manipula diretamente as impressoras, ou seja, é ele que envia os documentos para a impressora e é através dele que um documento pode ser removido da fila de impressão. O servidor Samba define apenas como vai ser realizado o acesso às impressoras.

Ferramentas de administração de servidores e serviços

Objetivos

Instalar, configurar e utilizar as ferramentas mais utilizadas para administração local e remota de servidores *web*, bem como manipular os serviços operativos.

Considerações iniciais

É muito comum, em servidores Linux, que o sistema instalado não possua interface gráfica, pois ela apenas significa consumo de recursos do computador, visto que os serviços instalados num servidor podem ser instalados, configurados e geridos através de comandos do terminal do Linux ou utilizando alguma aplicação que também esteja disponível apenas em modo texto. Outro motivo para que os sistemas Linux de servidores não possuam interface gráfica é que assim é possível evitar que outros utilizadores, além do próprio administrador da rede, utilizem o computador, evitando que lhe aconteça algum problema ocasionado por mau uso.

É comum também, em grandes redes, que exista mais de um computador que tenha a função de servidor. Isso acontece quando um serviço necessita de uma grande quantidade de processamento e opta-se por utilizar um computador apenas para aquele serviço. Dessa forma, pode acontecer existirem vários servidores (inclusive com sistemas operativos diferentes) numa mesma rede. Normalmente sem possuírem nem monitor, devido ao grande espaço que eles ocupam. Muitas empresas costumam adotar computadores especialmente desenvolvidos para operarem como servidores, pois são potentes, ocupam pouco espaço e foram desenvolvidos com o propósito de realizarem tarefas que necessitam de grande poder de processamento, tendo assim, um melhor desempenho. Eles ocupam pouco espaço físico pelo fato de que podem ser organizados em estantes (em inglês *racks*). A Figura 69 mostra um servidor da marca Dell.



Figura 69-Servidor Dell

Já a Figura 70 mostra uma estante com vários servidores instalados. Assim é possível colocar vários servidores em um pequeno espaço.



Figura 70-Estante de servidores Dell

Visto que os servidores não possuem monitores, o acesso é realizado remotamente. Isso significa que o servidor terá um serviço operante numa determinada porta, pela qual outros computadores poderão aceder ao servidor e geri-lo de várias formas, dependendo do sistema operativo instalado no servidor.

Em servidores com sistema operativo Linux, geralmente instala-se um servidor **SSH (Secure Shell)**. Através dele é possível ter acesso ao terminal do sistema do servidor e executar qualquer comando como se o utilizador estivesse utilizando o servidor *in loco*. Essa prática é muito comum, pois através do SSH, o servidor fica acessível a toda rede, desde que os utilizadores tenham permissão para acedê-lo. Algumas aplicações ainda oferecem uma interface gráfica que permite sua administração remotamente. Essa interface gráfica pode ser acessível através de um navegador, conforme o que se viu anteriormente ou pode existir uma aplicação específica que permita o acesso ao serviço.

Vai-se apresentar como fazer a instalação do serviço de SSH no servidor e como acedê-lo remotamente. Geralmente, o acesso ao servidor dá-se através de uma ferramenta de configuração e gestão que um serviço ofereça, seja ela via *browser* ou terminal de comandos.

Mostrar-se-a como aceder um servidor de bases de dados MySQL através da aplicação MySQL Administrator, que tem como objetivo oferecer uma alternativa ao PHPMyAdmin para a gestão das bases de dados.

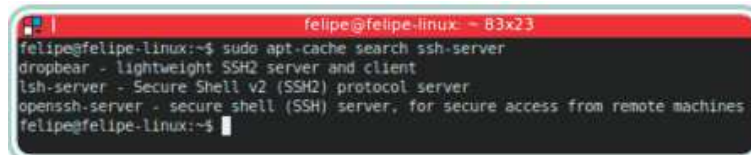
Instalação

Os serviços que oferecem interfaces gráficas acessíveis através do navegador já instalam essas interfaces automaticamente, na instalação da própria aplicação. Já ferramentas específicas devem ser instaladas manualmente. O mesmo acontece com o acesso via SSH, que necessita da instalação de um aplicativo no servidor que ficará monitorando uma porta, aguardando por ligações. Uma aplicação semelhante ao SSH é o **Telnet**, porém o seu uso só é aconselhável caso a ligação remota seja feita

numa rede muito segura, de preferência numa ligação direta com o servidor, pois o Telnet não oferece nenhum mecanismo de segurança durante a ligação, fazendo com que seja possível para um atacante obter informações de utilizadores e senhas da ligação. Já o SSH oferece uma ligação criptografada, permitindo que utilizadores utilizem a ligação com a *Internet* para aceder ao servidor com segurança.

Primeiramente, será abordada a instalação do servidor SSH e, posteriormente, a instalação do cliente de acesso à base de dados MySQL (MySQL Administrator).

A Figura 71 mostra os pacotes do servidor SSH disponíveis no repositório, através do comando `apt-cache search ssh-server` (`sudo apt-cache search ssh-server`).



```
felipe@felipe-linux: ~ 83x23
felipe@felipe-linux:~$ sudo apt-cache search ssh-server
dropbear - lightweight SSH2 server and client
lsh-server - Secure Shell v2 (SSH2) protocol server
openssh-server - secure shell (SSH) server, for secure access from remote machines
felipe@felipe-linux:~$
```

Figura 71-Pesquisa de pacotes do servidor SSH no repositório

Após verifi car os nomes dos pacotes, basta instalá-los através do comando `sudo apt-get install <nome-do-pacote>` (`sudo apt-get install openssh-erver`).

O **MySQL Admininstrator** é uma ferramenta presente no **MySQL Workbench**, que é um pacote de ferramentas gratuitas, tanto para sistemas Microsoft Windows quanto para sistemas Linux, que tem como objetivo oferecer meios gráficos para gestão de bases de dados. É possível obtê-lo através do website www.mysql.com na seção de ferramentas GUI (*Graphical User Interface*).

Configuração

A confi guração do servidor SSH consiste na alteração de parâmetros do arquivo de configuração `/etc/ssh/ssh_config`, que pode ser realizada editando o arquivo com qualquer editor de texto. Entre os vários parâmetros presentes no arquivo padrão, dois são considerados mais importantes, pois são mais úteis à personalização do serviço. A Figura 72 mostra esses parâmetros e a sua descrição.

Parâmetro	Descrição
port	Porta na qual o serviço irá operar. A porta padrão é a 22.
permitrootlogin	Indica se o usuário root poderá acessar o servidor via SSH. Caso essa opção seja definida como "no", e o usuário queira acesso root ao servidor, basta acessar com um usuário qualquer e, depois de conectado, executar os comandos utilizando o sudo.

Figura 72-Parâmetros de configuração do servidor SSH

Após a alteração dos parâmetros é necessário reiniciar o serviço do SSH, através do comando `sudo /etc/init.d/sshd restart` ou `sudo service sshd restart`.

Depois de ter realizado a configuração, basta aceder o servidor utilizando o comando **SSH <nome-de-utilizador>@ip-do-servidor**. Por exemplo, caso o nome do utilizador seja pedro e o endereço de IP do servidor seja 192.168.1.10, o acesso dá-se através do comando `SSH pedro@192.168.1.10`. Dessa forma, o acesso será realizado pela porta padrão. Caso o número da porta tenha sido alterado no parâmetro de configuração, ele deve ser definido no acesso, através do parâmetro **"-P numero-da-porta"**. Por exemplo, caso a porta definida tenha sido a 2000, o comando ficaria `SSH pedro@192.168.1.10 -P 2000`. Para aceder com outro utilizador, basta criá-lo no servidor.

Já o acesso ao servidor através de uma aplicação específica vai depender de cada aplicação, pois o acesso é definido nele e normalmente diferem um do outro. Será apresentado o acesso a um servidor

de base de dados MySQL apenas para exemplificar o processo. Ao abrir o MySQL Workbench, o utilizador irá encontrar várias maneiras de gerir a base de dados. A Figura 73 mostra a interface do MySQL Workbench no sistema Linux.

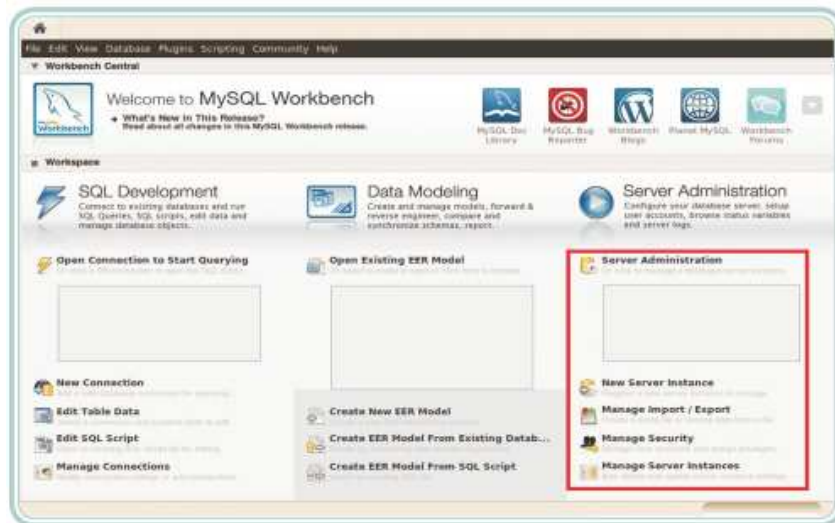


Figura 73-Ecrã inicial do MySQL Workbench

Na Figura 73, é possível notar, na parte destacada em vermelho, uma secção correspondente à administração do servidor, chamada **Server Administration**. Através dela é possível administrar o servidor. Clicando em **New Server Instance**, aparecerá um assistente que ajudará o utilizador a configurar uma ligação com a base de dados.

A Figura 74 mostra o MySQL Workbench já ligado ao servidor, onde no menu à esquerda, existem várias ferramentas de administração do servidor.

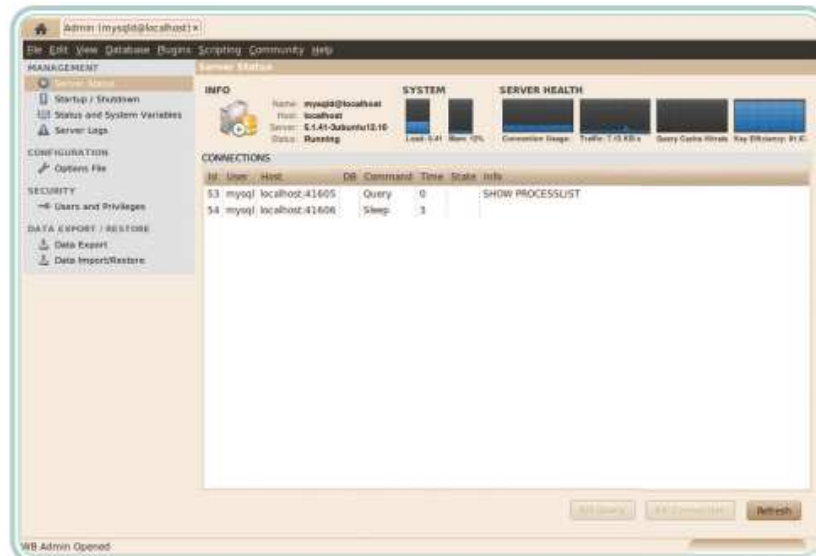


Figura 74-Mysql Workbench já ligado ao servidor

Resumo

Apresentou-se a instalação e configuração de um servidor SSH, uma ferramenta de grande auxílio para gestão remoto de servidores.

Mostrou, também, um exemplo de aplicação que oferece acesso remoto a um serviço presente no servidor. Nesse último caso, a ferramenta tem o propósito de gerir a aplicação em questão (MySQL, no caso). Para gerir o serviço, o acesso SSH é suficiente.

O serviço de acesso via SSH é muito importante em servidores, pois através dele o administrador é capaz de realizar qualquer comando no sistema, como se estivesse operando o servidor *in loco*. É possível iniciar, reiniciar e parar serviços, da mesma forma que alterar arquivos de configuração.

Já em casos específicos, podem existir ferramentas desenvolvidas especialmente para gerir um tipo de serviço apenas, como é o caso do MySQL Administrator. Através dele é possível gerir o serviço do servidor MySQL, da mesma forma que gerir as tabelas presentes na base de dados, realizar consultas, entre outras funcionalidades.

Referências

APACHE SOFTWARE FOUNDATION. Apache HTTP server version 2.2 documentation.
Disponível em: <<http://httpd.apache.org/docs/2.2/>>.

DNSMASQ. dnsmasq manual.

Disponível em: <<http://thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html>>

Fundamentos da arquitetura cliente/servidor.

Disponível em: http://www.unlu.edu.ar/~tyr/tyr/TYR-anterior/Fundamentos_da_%20Arquitetura_Cliente-Servidor.pdf

Local DNS cache for faster browsing on Ubuntu machine.

Disponível em: <<http://www.ubuntugeek.com/local-dns-cache-for-faster-browsing-on-ubuntu-machine.html>>

POSTFIX. Postfix basic configuration.

Disponível em: <http://www.postfix.org/BASIC_CONFIGURATION_README.html>