sqtpm

Guilherme P. Telles

1º de fevereiro de 2015

O sqtpm é um sistema para recepção via web, compilação e verificação de correção de trabalhos de programação. Permite a recepção de programas em C, C++, Fortran e Pascal, e de arquivos PDF, sozinhos ou acompanhando programas.

A primeira versão do sistema foi escrita em 2003 quando assumi duas turmas de programação na USP. Organizar e verificar a correção dos trabalhos de programação consumia muito tempo. O sistema evoluiu um pouco com o passar do tempo, mas a filosofia continua a mesma desde a primeira versão: o sistema é configurado e organiza os trabalhos por diretórios, arquivos e links. O responsável pelos trabalhos gerencia o sistema na linha de comandos. Não há uma interface web de administração, nem há uma interface específica para avaliação de aspectos não-funcionais dos trabalhos.

O sqtpm teve sua inspiração inicial no sistema MC102 construído pelo Zanoni Dias na Unicamp e principalmente nas conversas com o próprio Zanoni, mas não tem código do MC102. Não obstante, a forma de limitar os recursos via `ulimit` durante a execução dos programas submetidos é igual à usada pelo MC102. À época da primeira versão também tomei conhecimento acerca da existência de outros sistemas do gênero, mas independentemente da existência deles achei melhor fazer do zero do que usar ou adaptar outros sistemas. Desta forma eu teria o prazer de programar, poderia tomar as decisões que eu bem entendesse e não precisaria lidar com programas de outras pessoas.

Este documento descreve o sqtpm. Foi escrito inicialmente para amadurecer as idéias, antes de programar, e agora tem servido para ajudar a manter, a instalar e a configurar o sistema. É uma mistura de documento de requisitos, de arquitetura e manual, que não é nenhum deles. O sistema foi colocado no ar pela primeira vez em janeiro de 2004 e a versão descrita neste documento é a 6. O sqtpm é distribuído nos termos da WTFPL v2.

Conteúdo

1 Organização do sistema

O sqtpm está escrito em Perl. Usa Session de CGI e ExpireSessions de Session, gcc, make, gpc, diff e opcionalmente indent. Também empacota o google-code-pretiffier. O sistema nunca foi testado em um sistema diferente do GNU/Linux com Apache.

O sqtpm é organizado em uma hierarquia de diretórios onde os componentes dele, as definições de trabalhos e os programas enviados ficam armazenados. O sistema funciona em um diretório que chamaremos **diretório-raiz**. O diretório-raiz contém:

- os programas e arquivos do sistema,
- os arquivos de usuários, com sufixo `.pass` e
- um subdiretório para cada trabalho.

1.1 Usuários e senhas

No diretório-raiz deve haver um ou mais arquivos de usuários com nome que tenha sufixo `.pass`. O formato de um arquivo de usuários é:

```
identificador:senha-criptografada
identificador:senha-criptografada
...
identificador:senha-criptografada
```

Os arquivos de usuários devem ser criados apenas com os identificadores, um por linha, ou com identificadores seguidos de `:`, um por linha. O caractere `#` faz com que o conteúdo de uma linha seja ignorado a partir da primeira ocorrência dele.

Os identificadores de usuários são cadeias de letras e dígitos, opcionalmente precedidos por um asterisco. Cada identificador deve ser único na união de todos os arquivos de senhas.

O sistema tem dois tipos de usuários: professor e aluno. Usuários que têm um asterisco precedendo o identificador no arquivo de senhas são do tipo professor. Um professor pode ver relatórios consolidados com todas as submissões para cada turma e pode submeter trabalhos antes do início do prazo ou depois do fim prazo, sem limite no número de submissões. Usuários que não têm uma asterisco precedendo o identificador são alunos. Um aluno pode ver apenas trabalhos e envios dele mesmo.

Cada usuário cadastra sua senha sem intervenção, através da interface do sistema. O sistema foi construído supondo que a lista de usuários possa ser extraída facilmente de algum sistema acadêmico e que cada arquivo contenha usuários de uma mesma turma, ou de outro grupo que faça sentido nas disciplinas.

1.2 Trabalhos

Todo subdiretório do diretório-raiz cujo nome não começa com ponto ou com sublinhado e que contém um arquivo chamado `config` é um trabalho. O nome do diretório de um trabalho não deve ter espaços. No diretório de um trabalho ficam:

- o arquivo de configuração do trabalho, chamado `config`,
- links simbólicos para arquivos de usuários,
- um diretório para cada usuário que enviou o trabalho,
- opcionalmente, os casos-de-teste do trabalho,
- opcionalmente o programa verificador,
- opcionalmente, o diretório `extra-files`,
- opcionalmente, o arquivo `casos-de-teste.tgz`,

- opcionalmente, arquivos `.html` e `.png` que compõem o enunciado e
- opcionalmente, o diretório `backup`.

Ao submeter um trabalho, um usuário deve enviar um ou mais programas-fonte. O trabalho pode ser configurado para receber arquivos PDF acompanhando os fontes e para limitar a quantidade e nomes dos arquivos.

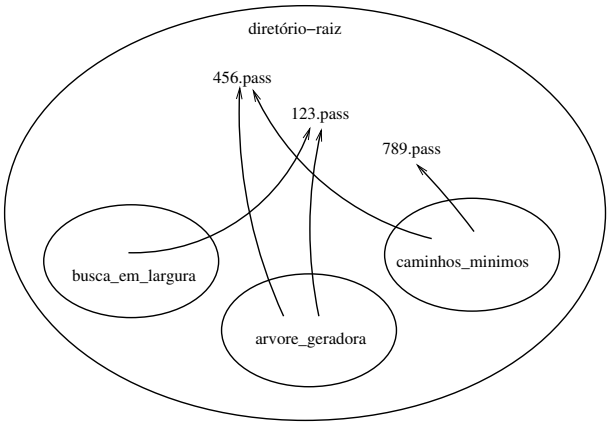
1.2.1 Permissão para submeter

Um usuário enxerga e pode enviar um certo trabalho `T` se o identificador dele estiver em algum arquivo de usuários para o qual existe um link dentro do diretório `T`. Essa estrutura permite administrar as autorizações através da criação e remoção de links simbólicos.

Por exemplo, suponha que há três turmas de alunos que submeterão trabalhos da seguinte forma:

Trabalho	Turmas que devem fazer o trabalho
busca em largura	123
árvore geradora	123, 456
caminhos mínimos	456, 789

Os arquivos de usuários e respectivos links simbólicos podem ser construídos da forma como aparecem na figura abaixo.



1.2.2 Enunciado

O sistema pode exibir o enunciado de cada trabalho. O enunciado pode ser um arquivo html dentro do diretório do trabalho ou pode ser uma url. Se for um arquivo html, o sistema exibe o arquivo. Se for uma url, o sistema exibe um link para ele. O sistema também exibe informações sobre o trabalho, como datas, linguagens, número e nomes de arquivos e outras.

1.2.3 Download de casos-de-teste

Se houver um arquivo chamado `casos-de-teste.tgz` no diretório do trabalho então o sistema exibe um link para download desse arquivo juntamente com o enunciado.

2 Envio, compilação e execução

Quando um usuário envia um trabalho, o sistema verifica o identificador, a senha, o número máximo de submissões, as datas-limite, a linguagem, a quantidade, os nomes e os sufixos dos nomes dos arquivos. Se o usuário com identificador `id` submete o trabalho de programação `T` então depois das verificações os arquivos enviados são gravados no diretório `raiz/T/id/`.

Quando o trabalho é apenas para recepção de arquivos PDF então o sistema grava os arquivos no diretório do usuário e termina, sem emissão de uma pontuação. Caso contrário, o programa é compilado e executado para cada um dos casos-de-teste que estiverem no diretório do trabalho. Depois de executados, os resultados dos casos-de-teste são verificados e o usuário recebe uma pontuação de 0 a 100%, descontada da multa por atraso se houver. A pontuação pode ser proporcional ao número de casos-de-teste bem sucedidos ou pode ser "0 ou 100". Se não houver casos-de-teste, o sistema recebe o programa e compila. A pontuação será 100% descontada da multa por atraso se a compilação for bem sucedida ou 0 caso contrário.

Se houver um diretório `extra-files` então todos os arquivos e subdiretórios dele serão copiados para o mesmo diretório em que o executável estará durante a execução dos casos-de-teste.

Se o usuário fizer uma re-submissão do trabalho `T` e se a opção de backup estiver ativa, os arquivos pré-existentes serão movidos para o diretório `raiz/T/backup/id.tentativa.data/`, onde `tentativa` é o número daquele envio e `data` é a data em que o programa substituído havia sido enviado. Caso contrário, os arquivos pré-existentes serão removidos.

2.1 Verificação e casos-de-teste

A forma dos casos-de-teste depende do tipo de verificação que será realizada. Há duas formas de verificação de acerto na execução de um caso-de-teste: por comparação com uma saída esperada e por programa verificador. A seleção do tipo de correção é feita no arquivo de configuração do trabalho.

Verificação por comparação Se a verificação for por comparação com uma saída esperada, cada caso-de-teste `z` é composto por dois arquivos:

- `z.in`, com a entrada para o programa e

- `z.out`, com a saída esperada do programa quando alimentado com a entrada `z.in`.

O resultado da execução de cada caso-de-teste é comparado com o esperado usando `diff`. O resultado pode ser:

- `bem-sucedido`: o resultado produzido pelo programa está idêntico ao esperado.
- `saída com formatação incorreta`: o resultado produzido pelo programa difere do esperado por espaços, fins-de-linha ou caixa de caracteres. O caso-de-teste é considerado mal-sucedido.
- `saída incorreta`: o resultado produzido pelo programa difere do esperado. O caso-de-teste é considerado mal-sucedido.
- `limite de tempo ou memória excedido`.
- `violação de memória`.
- `erro de ponto flutuante`.
- `erro de execução (n)`: erro de execução com exit status n diferente de 9, 11 ou 8.

Embora o nome do arquivo de um caso-de-teste possa ser uma string arbitrária, a tela do sistema exibe apenas números de 1 até n que correspondem à ordem lexicográfica dos nomes dos arquivos. Uma boa prática de organização tem sido dar nomes da forma `dd-string.{in,out}`, porque permitem uma associação direta entre o nome exibido na tela e o nome do arquivo. Por exemplo, `01-grafo-vazio.in`, `02-k5.in`, `03-k3-3.in` etc.

Verificação por programa A verificação através de programa é essencial quando a resposta correta para uma entrada não for única.

Se a verificação for feita por um programa, então cada caso-de-teste deverá ser composto apenas de um arquivo de entrada `z.in`. O resultado da execução será igual ao produzido na verificação por comparação e dependerá do valor retornado pelo programa verificador, como discutido na Seção 4.2.

2.2 Log de submissões

O sistema grava um log de submissões chamado `sqtpm.log` no diretório-raiz. Cada linha do log refere-se a uma submissão ou tentativa de acesso.

3 Linguagens e compiladores

As linguagens de programação suportadas pelo sistema são C com gcc, C++ com g++, Fortran com gfortran e Pascal com gpc. O acoplamento entre os compiladores e o sqtpm é moderadamente forte, de forma que a troca de algum compilador provavelmente vai exigir modificar o sqtpm.

As opções globais para os compiladores são definidas no `sqtpm.cfg`. Elas podem ser redefinidas para cada trabalho, como está descrito na Seção 4.2. Não haverá problema em não ter um compilador instalado se a linguagem não for usada.

Programas em C e C++ podem ser formados por um ou mais arquivos. Fortran e Pascal estão limitadas a receber apenas um arquivo-fonte porque as versões mais antigas do sqtpm tinham essa limitação e não fiz os ajustes necessários para removê-las. Essa limitação está cravada no código do sqtpm.

Os nomes dos arquivos-fonte devem ser cadeias de letras, dígitos, pontos, hífens e sublinhados com os seguintes sufixos, de acordo com a linguagem:

- C: .c e .h
- C++: .cpp e .h
- Pascal: .pas
- Fortran: .f ou .F

Há ainda a "linguagem" PDF, que apenas permite a recepção de arquivos desse tipo. Para esses arquivos o sufixo deve ser `.pdf`.

4 Configuração

Há um arquivo para configuração do sistema e cada trabalho deve ter seu próprio arquivo de configurações.

Tais arquivos contêm linhas da forma `diretriz=valor`. O caractere `#` faz com que o conteúdo de uma linha seja ignorado a partir de sua primeira ocorrência.

4.1 Configuração do sistema

O arquivo `raiz/sqtpm.cfg` define valores-padrão de diretrizes para todos os trabalhos e diretrizes de configuração do sistema.

Um modelo de arquivo aparece na Figura 1. As diretrizes mostradas no modelo não devem ser omitidas. A próxima seção inclui uma descrição das primeiras doze. As demais especificam o path para programas externos.

Diretrizes que aparecem na configuração do sistema podem ser redefinidas na configuração de cada trabalho. Embora as cinco últimas diretrizes no exemplo da Figura 1 possam ser redefinidas na configuração de trabalhos isso não foi testado.

4.2 Configuração de trabalhos

Cada diretório de trabalho deve conter um arquivo chamado `config`. As diretrizes definidas nesse arquivo afetam apenas o trabalho para o qual foram definidas. As diretrizes válidas são as seguintes.

```
# Configuracoes para o sqtpm.
# Valores-padrao de diretrizes para trabalhos:
languages = C C++ Fortran Pascal PDF
scoring = total
penalty = 100
backup = on
tries = 25

cputime = 1
virtmem = 32768
stkmem = 8192

gcc-args = -Wall -O3
g++-args = -Wall -O3
gfortran-args = -Wall
gpc-args = -Wall -O2

# Paths:
gcc = /usr/bin/gcc
g++ = /usr/bin/g++
gfortran = /usr/bin/gfortran
gpc = /usr/local/bin/gpc -Wa,--32 -B /usr/lib32
make = /usr/bin/make
diff = /usr/bin/diff
indent = /usr/bin/indent
```

Figura 1: Um exemplo de arquivo `sqtpm.cfg`.

- `languages = lista` Define uma lista de linguagens de programação que serão aceitas. Os itens da lista devem ser separados por espaços. Os valores válidos são: `C C++ Fortran Pascal PDF`.
- `deadline = aaaa/mm/dd hh:mm:ss` Define a data e o horário limites para a submissão do trabalho. Se não for incluída, o trabalho não deixa de ser aceito. O formato deve ser seguido estritamente.
- `startup = aaaa/mm/dd hh:mm:ss` Define a data e o horário para começar a exibir o enunciado e passar a aceitar submissões de um programa. Se não for incluída, o trabalho passa a ser exibido para usuários que fizerem login depois que os links para arquivos de usuários forem criados. O formato deve ser seguido estritamente.
- `penalty = n` Define a multa percentual sobre os pontos obtidos no trabalho, cobrada por dia de atraso passado da data limite para submissão. Deve ser um número inteiro no intervalo $[0, 100]$. A multa passa a ser aplicada já no primeiro segundo após a data limite. O sistema continuará aceitando o trabalho enquanto o usuário puder ter pontuação maior que zero.
- `scoring =` {`total`,`proportional`} Define a forma de calcular a pontuação para o trabalho. `total` significa pontuação 100% se o programa for bem sucedido em todos os casos-de-teste ou pontuação 0% caso contrário. `proportional` significa proporcional ao número de casos-de-teste bem sucedidos.

- `description = string` Define o enunciado do trabalho. Há duas alternativas. A primeira é o nome de um arquivo html no diretório do trabalho, que será ecoado no browser. A outra é uma url iniciada por `http` para a qual o sqtpm exibirá um link. Exemplos são:

  ```
  description = http://143.107.183.131/problemas/t1.html
  description = enunciado.html
  ```

- `backup = {on,off}` Define se o sistema manterá ou não cópias das submissões anteriores.
- `cputime = n` Define o tempo de CPU máximo para executar cada caso-de-teste, em segundos (número inteiro). Esse valor é usado com `ulimit -t`.
- `virtmem = n` Define o tamanho máximo de memória virtual para executar cada caso-de-teste, em kilobytes (número inteiro). Esse valor é usado com `ulimit -v`.
- `stkmem = n` Define o tamanho máximo de pilha para executar cada caso-de-teste, em kilobytes (número inteiro). Esse valor é usado com `ulimit -s`.
- `showcases = lista` Define uma lista de nomes de casos-de-teste para os quais o sistema mostrará a entrada, a saída esperada e a saída produzida pelo programa enviado. Se a correção for feita por programa verificador, então não haverá a exibição da saída esperada. Por exemplo:

  ```
  showcases = 01-grafo-vazio 03-k3-3
  ```

- `tries = n` Define o número máximo de vezes que um trabalho pode ser enviado.
- `filenames = lista` Define uma lista de nomes separados por espaços. Se for definida então a submissão de um trabalho deve incluir arquivos com os nomes na lista. Se a cadeia {`uid`} fizer parte de algum nome, ela será substituída pelo identificador do usuário. Se a cadeia {`assign`} fizer parte de algum nome, então ela será substituída pelo nome do trabalho.
- `sources = n,m` Se esta diretriz for definida com valores inteiros `n` e `m` tais que $0 < \texttt{n} \leq \texttt{m}$ então o sistema exige que o número de arquivos fonte esteja entre `n` e `m`. Esta diretriz não afeta a linguagem PDF.
- `pdfs = n,m` Se esta diretriz for definida com valores inteiros `n` e `m` tais que $0 < \texttt{n} \leq \texttt{m}$ então o sistema exige que o número de arquivos PDF esteja entre `n` e `m`. Esta diretriz afeta todas as linguagens.

- `verifier = comando` Define um comando para a execução de um programa verificador. Se esta diretriz não estiver presente, o sistema fará a verificação dos casos-de-teste por comparação da saída.

  O programa verificador deve aceitar como parâmetros dois nomes de arquivos: a entrada de um caso-de-teste e a saída produzida pelo programa (ambos com path absoluto), nesta ordem. O programa verificador deve retornar 0 se a saída for correta, 1 se for incorreta, 2 se houver erro de formatação e > 2 se houver um erro de execução do próprio verificador. O programa verificador não será interrompido pelo sqtpm e portanto deve ser o mais robusto possível para que não surjam zumbis.

  Se o comando começar com o caractere `@`, então o caractere `@` será substituído pelo path do diretório do trabalho. Por exemplo, se um trabalho `T` possui como verificador um executável chamado `verif`, que está no próprio diretório do trabalho, a diretriz será:

  ```
  verifier = @verif
  ```

  Outros exemplos são:

  ```
  verifier = /tmp/verif
  verifier = /usr/bin/perl @verif.pl
  verifier = /usr/bin/perl /tmp/verif.pl
  ```

- `gcc-args = string`
  `g++-args = string`
  `gpc-args = string`
  `gfortran-args = string`

  Definem parâmetros adicionais para os compiladores. Não devem incluir o parâmetro `-o`.

4.3 Múltiplas instâncias no mesmo servidor

Caso seja necessário ter mais de uma instância no mesmo servidor então o `sqtpm.cgi` tem que ser editado para que cada instância tenha nomes distintos para os arquivos de sessões. O padrão é usar o prefixo `sqtpm-sess-` em `/tmp/`.

5 Relatórios

Os relatórios de envio podem ser vistos pelos alunos que os enviaram e pelos professores de uma turma.

Professores vêem as notas de todos os usuários tabuladas por arquivo de usuários. Para cada submissão um link permite a exibição do relatório da submissão e dos arquivos que foram enviados. Se o arquivo enviado for um programa então a sintaxe será destacada pelo google-code-prettifier. Programas C também serão endentados usando indent. Arquivos PDF ficam disponíveis para download.

6 Integridade e execução dos casos-de-teste

Para evitar que alguém explore os diretórios dos trabalhos diretamente e veja os programas enviados por outras pessoas e os arquivos de configuração do sqtpm, o servidor http deve ser configurado para bloquear listagem de diretórios e bloquear acesso a qualquer arquivo que não seja `.cgi`, `.js` e `.css`. Essa restrição deve ser aplicada ao diretório-raiz e a todos os subdiretórios dele. Também é interessante usar alguma ferramenta para filtrar ataques ao servidor.

Executar código alienígena é uma brecha de segurança enorme. O sqtpm executa cada caso-de-teste através de um wrapper chamado `sqtpm-etc.sh` que recebe como parâmetros o identificador do usuário, o nome do trabalho, o tempo máximo de CPU, o tamanho da memória virtual e o tamanho da pilha. O wrapper deve executar cada caso-de-teste aplicando as restrições de memória e tempo de CPU. Também deve fazer com que o conteúdo do diretório `extra-files` esteja no mesmo lugar onde o executável estiver, se ele existir. Para cada arquivo `z.in` esse wrapper deve escrever os seguintes arquivos no diretório `raiz/trabalho/_uid_tmp_/`:

- `z.run.out`: stdout da execução do caso-de-teste.
- `z.run.err`: stderr da execução do caso-de-teste.
- `z.run.st`: exit status da execução do caso-de-teste.

O wrapper mais simples e insano executa os casos-de-teste no próprio servidor. Esse é o `sqtpm-etc-localhost.sh`, que criei apenas para testar. Soluções melhores usam uma jaula ou uma máquina virtual.

Criei o par `sqtpm-etc-vbox-shared.sh` e `vbox-etc-shared.sh` para usar com uma maquina-virtual no VirtualBox com diretório compartilhado. Testei também um par de scripts para VirtualBox sem diretório compartilhado e é bem mais lento.

Sempre que houver uma tentativa de enviar uma requisição http com parâmetros incorretos, por exemplo para um trabalho para o qual o usuário não tem permissão, a sessão do usuário será terminada e ele será bloqueado. O arquivo `.pass` que contém o usuário será alterado e o sistema vai adicionar uma linha ao log.

No segundo semestre de 2013 houve um número muito grande de submissões por alguns alunos (na casa das centenas), o que é um tipo de ataque.

```
-rwxr-x---   1 gpt  www-data  sqtpm.cgi
-rwxr-x---   1 gpt  www-data  sqtpm-pwd.cgi
-rw-r-----   1 gpt  www-data  sqtpm.pm
-rw-r-----   1 gpt  www-data  sqtpm.cfg
-rw-r-----   1 gpt  www-data  sqtpm.css
-rw-r-----   1 gpt  www-data  sqtpm.js
-rw-rw----   1 gpt  www-data  sqtpm.log

-rw-r-----   1 gpt  www-data  bula.html
-rw-r-----   1 gpt  www-data  envio.html
-rw-r-----   1 gpt  www-data  saida.html
-rw-r-----   1 gpt  www-data  icon.png
-rw-r-----   1 gpt  www-data  keep-calm-and-0-no-sqtpm-200.png

drwxr-s---   2 gpt  www-data  google-code-prettify

lrwxrwxrwx   1 gpt  www-data  sqtpm-etc.sh -> sqtpm-etc-vbox-shared.sh
-rwxr-x---   1 gpt  www-data  sqtpm-etc-vbox-shared.sh
-rwxr-x---   1 gpt  www-data  sqtpm-etc-vbox-noshared.sh
-rwxr-x---   1 gpt  www-data  sqtpm-etc-localhost.sh

drwxrws---   2 gpt  www-data  conta
```

Figura 2: Arquivos e diretórios do sqtpm.

Então implementei um limite do número de submissões. Como opção considerei também uma espera à medida que o número de submissões aumenta, mas gostei mais do limite.

7 Arquivos e permissões

Os arquivos e diretórios que compõem o sqtpm estão listados na Figura 2. Diretórios de trabalhos devem ter modo 2770, como o do trabalho `conta`, que serve para testar a instalação.

Ainda há os seguintes arquivos de apoio (diretório Utils):

- `fix-perms.sh` Script que ajusta as permissões dos arquivos do sistema e de todos os trabalhos (quando executado sem parâmetros) ou apenas para um trabalho (quando executado com o nome do trabalho como parâmetro).
- `sqtpm-clean-sessions.sh` Script para remover os arquivos de sessões expiradas.
- `sqtpm-vbox` Script init para a VM do sqtpm.
- `sqtpm-vbox-pause.sh` Script para a crontab do usuário `www-data`, que coloca a VM em pausa se não houver alguma sessão aberta, o que reduz o uso de CPU pela VM. Invoca o `sqtpm-clean-sessions.sh`.

8 Instalação

Nesta seção estão descritos os passos que usei para instalar o sqtpm no Debian 7.7.0 com execução dos programas enviados em uma máquina virtual com diretório compartilhado.

Instalei em `/home/www/sqtpm/`, com `www-apache:www-apache` e `2770`. com o seguinte ambiente: Perl (v5.14.2) com CGI::Session e CGI::Session::ExpireSessions, Apache (Apache/2.2.22 (Debian)), VirtualBox (4.1.18_Debian r78361), gcc (Debian 4.7.2-5), g++ (Debian 4.7.2-5), GNU Fortran (Debian 4.7.2-5), gpc (20041218, based on gcc-3.3.3) e GNU Make 3.81.

1. Acrescentei o seguinte em `/etc/apache2/sites-available/default`.

```
Alias /sqtpm/ /home/www/sqtpm/

<Directory /home/www/sqtpm>
  AddHandler cgi-script .cgi
  Options +ExecCGI -MultiViews
  AllowOverride None

  <FilesMatch "\.(cgi|js|css|png)$">
    Order allow,deny
    Allow from all
  </FilesMatch>

  Order allow,deny
  Deny from all
</Directory>
```

2. Criei uma partição pequena (alguns poucos gigas) no meu sistema e montei em `/mnt/aux/`. Setei 777. Incluí no fstab.
3. Criei uma VM usando o Debian live 7.7.0 standard, com 512 MB de RAM, uma CPU e 2 GB de disco. Durante a instalação criei uma boa senha de root e o usuário sqtpm. Fiz login como root e desabilitei a rede e outros serviços desnecessários como bluetooth, impressão, servidor de email etc, limpei as crontabs etc. Instalei o extension pack e criei um diretório compartilhando `/mnt/aux/` como `/media/sf_aux/`, com auto-mount e permanent. Desabilitei a rede e a USB no controle da VM. Copiei o script `vbox-etc-shared` em `/home/sqtpm/` como root:root e 0755, mudei os arquivos do usuário sqtpm para root:root e 644. Depois exportei essa VM.
4. Importei a VM como www-data. (Meu sistema tem o `/` pequeno então primeiro mudei o homedir de www-data para `/home/www-data/`.)
5. Instalei o script init para a VM.
6. Criei `/home/www/` e `/home/www/sqtpm/` com www-data:www-data e 2770.

7. Copiei os arquivos do sistema em `/home/www/sqtpm`, criei um link simbólico de `sqtpm-etc.sh` para `sqtpm-etc-vbox-shared` e executei `fix-perms.sh` como root.
8. Criei um arquivo de senhas com um usuário, linkei no `conta` para testar.
9. Reiniciei o apache, subi a VM e pronto.