Trabalho prático de Sistemas Operativos
Backup Eficiente

Grupo de Sistemas Distribuídos
Universidade do Minho

Abril de 2016

Informações gerais

- O trabalho é realizado em grupo, sendo estes constituídos por até três elementos. Grupos de dimensão inferior a três terão de ser justificados e autorizados pelos docentes. A constituição de cada grupo tem de ser registada no Blackboard até ao dia 22 de Abril. É também nesta altura que os alunos que pretendam manter a nota do trabalho do ano passado ($<= 15$ valores) se devem registar no grupo especial "quero manter a nota de 2015".
- O trabalho deve ser entregue até às 23:59:59 do dia 20 de Maio de 2016.
- Para permitir a divulgação na nota do trabalho antes do início dos testes, a apresentação deverá ter lugar nos dias 23 a 25 de Maio, com inscrição via Blackboard.

Resumo

Neste trabalho pretende-se construir um sistema eficiente de cópias de segurança (*backup*), para salvaguardar ficheiros de um utilizador. Estes sistemas têm duas operações principais, a serem invocadas pelo utilizador:

Backup : cria a cópia dos ficheiros/pastas indicados

Restore : restaura ficheiros/pastas, repondo o backup para o local original

À primeira vista, parece uma tarefa simples, dir-se-ia "de primeiro ano". Bastaria um comando ou script em Bash para copiar numa ou noutra direcção. No entanto, neste trabalho vamos colocar requisitos adicionais, designadamente de eficiência e privacidade, conduzindo a uma arquitectura com vários processos concorrentes e a soluções semelhantes às estudadas nas aulas práticas de Sistemas Operativos.

Eficiência : é necessário minimizar o espaço em disco ocupado pelo backup; para isso usa-se compressão de dados (eliminação da redundância dentro de um ficheiro) e deduplicação (eliminação de ficheiros duplicados)

Privacidade : é necessário usar uma arquitectura cliente/servidor, impedindo o acesso directo do utilizador à pasta/dispositivo de backup

Este sistema deverá então ser composto por dois programas: um programa cliente que serve para dar comandos ao sistema e um programa servidor que efetua as operações de cópia e reposição propriamente ditas.

Utilização

O servidor é iniciado da seguinte forma:

```
$ sobusrv
```

Depois de iniciado, o servidor deve permanecer em execução, aguardando pedidos do cliente para operações de cópia e recuperação de ficheiros. O cliente pode ser usado para efetuar uma cópia de vários ficheiros da seguinte forma:

```
$ sobucli backup *.txt
a.txt: copiado
b.txt: copiado
$ _
```

Para recuperar um dos ficheiros previamente guardados, usa-se o cliente da seguinte forma:

```
$ sobucli restore a.txt
a.txt: recuperado
$ _
```

Em qualquer dos casos, o cliente espera pela confirmação do servidor que cada ficheiro está efetivamente copiado ou recuperado antes de imprimir a mensagem correspondente. O cliente só termina quando todas as operações solicitadas estiverem concluídas.

Implementação

A implementação deve cumprir os seguintes requisitos:

- O sistema deve ser implementado em C usando as primitivas do sistema operativo e programas utilitários Unix indicados.
- O servidor (`sobusrv`) guarda todos os dados numa única diretoria. A esta diretoria chamamos a *raiz do backup*. Sugere-se que seja usada a diretoria `/home/`*user*`/.Backup/`.
- O cliente (`sobucli`) nunca escreve nem lê ficheiros diretamente na raiz do backup. O cliente envia comandos para copiar e recuperar ficheiros ao servidor usando um FIFO / *named pipe* que se encontra na raiz do *backup*.
- O servidor envia respostas ao cliente, para indicar a conclusão das operações ou erros, usando apenas sinais.
- Deve ser possível executar concorrentemente várias instâncias do programa cliente sem que isso cause problemas.
- O servidor nunca deve executar mais do 5 operações (de cópia ou recuperação) em simultâneo, de forma a não sobrecarregar o sistema.

O armazenamento dos ficheiros guardados em duas sub-diretorias da raiz do backup com os nomes `data` e `metadata` da seguinte forma:

- O conteúdo do ficheiro deve ser comprimido com o programa `gzip` e guardado na sub-diretoria `/data/` num ficheiro cujo nome é o *digest* do seu conteúdo não comprimido calculado pelo programa `sha1sum`.

- Deve ser criado na sub-diretoria `metadata/` uma ligação com o nome original do ficheiro para o seu conteudo comprimido na sub-diretoria `data/`.

Considere o seguinte exemplo: Admita que tem dois ficheiros dos quais calculamos o *digest*. Se efetuarmos cópias de segurança de ambos, ficamos com o conteúdo seguinte na sub-diretoria `data/`:

```
bc38e2f249dbf7c74eaa823f65ddda9a122013f3
fda670a8d9bd15a411245324be40d745de278d75
```

em que o conteúdo de `bc38e2f249dbf7c74eaa823f65ddda9a122013f3` é o de `a.pdf` depois de comprimido com `gzip`. O outro corresponde da mesma forma a `b.pdf`. O conteúdo da diretoria `metadata/` é o seguinte:

```
a.pdf -> ../data/bc38e2f249dbf7c74eaa823f65ddda9a122013f3
b.pdf -> ../data/fda670a8d9bd15a411245324be40d745de278d75
```

Desta forma conseguimos recuperar o conteúdo de um ficheiro procurando-o em `metadata/` e descomprimindo o conteúdo correspondente em `data/`.

Se for feita uma cópia de segurança de um `c.pdf` com o mesmo conteúdo de `a.pdf`, como descobrimos pelo *digest* que os dados são os mesmos de um ficheiro que já temos guardado, não é preciso guardar nova cópia. Basta acrescentar a `metadata` a entrada correspondente, obtendo:

```
a.pdf -> ../data/bc38e2f249dbf7c74eaa823f65ddda9a122013f3
b.pdf -> ../data/fda670a8d9bd15a411245324be40d745de278d75
c.pdf -> ../data/bc38e2f249dbf7c74eaa823f65ddda9a122013f3
```

Valorização

Fica ao critério de cada grupo apresentar uma solução mais sofisticada. Por exemplo, o trabalho é valorizado pelas seguintes funcionalidades adicionais:

- Suporte para um comando `sobucli delete` para remover um ficheiro da cópia de segurança. Atenção que os seus dados podem ser partilhados por vários ficheiros!
- Suporte de executar `sobucli gc` para eliminar todos os ficheiros na sub-diretoria `data/` que não estejam a ser usados por nenhuma das entradas em `metadata/`.
- Possibilidade de indicar uma diretoria base para cópia ou recuperação em vez de ficheiros isolados. Considere o programa `find` para o ajudar a implementar esta possibilidade.
- Possibilidade de operação simultânea por parte de vários utilizadores. Crie contas para todos os elementos do grupo num dos computadores e peça aos colegas de grupo para fazerem login remoto nesse computador e experimentarem fazer backups de ficheiros dos 3 elementos em simultâneo. Sem quebrar o requisito da privacidade, obviamente...

Notas finais

Todos os ficheiros (código, makefile, script de instalação, relatório em pdf) devem ser comprimidos num único ficheiro ZIP e submetidos via Blackboard até à hora indicada, sendo penalizadas as submissões tardias. Para esse efeito, em breve serão abertos no Blackboard muitos (100?) grupos com capacidade de 3 elementos, devendo cada grupo escolher um grupo vazio e preencher o nome dos elementos desse

grupo[1]. Uma vez conhecido o número de grupos que submeteram o trabalho e feita a divisão pelos docentes disponíveis, serão criados ”slots” para o horário de apresentação nas datas anunciadas, onde um dos elementos deverá inserir o número do grupo. Dessa forma, basta que cada grupo compareça à hora indicada, ficando com o resto do dia livre.

Este trabalho vale apenas 1/3 da classificação final. É relativamente simples para quem foi resolvendo os exercícios propostos nos guiões e serve essencialmente para averiguar o grau de preparação para o teste e/ou exame de recurso, onde não existe ajuda dos colegas.

Todos os elementos do grupo terão de estar à vontade para responder às questões durante a discussão pública, justificando as opções tomadas e mostrando conhecimento do código desenvolvido, dos testes efectuados, da configuração do sistema, etc. Pretende-se que cada um mostre o que aprendeu e que estará à vontade no teste/exame. Assegure-se que percebeu bem que problema resolve cada solução que implementou: que problema resolve este fork aqui, este pipeline, este wait, estes sinais. . . .

Não tente copiar ideias de outros grupos e evite a todo o custo usar funções ou comandos que lhe tiram “muito trabalho”, a começar pelo `system`! Mostre o que sabe fazer e porque o fez.

Note que é muito provável que, após serem submetidos no Blackboard, os trabalhos sejam copiados para uma máquina virtual linux e no dia da avaliação seja pedido a *um ou mais* elementos do grupo para demonstrarem o trabalho nessa máquina virtual (sem ajuda dos colegas).

[1] Caso vejam um grupo parcialmente preenchido, farão o favor de escolher um grupo vazio!