Exercícios sobre a Ferramenta Dtrace

Sérgio Caldas
Universidade do Minho
Escola de Engenharia
Departamento de Informática
Email: a57779@alunos.uminho.pt

***Resumo*—Este relatório, exprime os resultados obtidos na resolução de exercícios sobre *Dtrace*, no âmbito da disciplina de Engenharia de Sistemas de Comuptação (ESC), inserida no perfil de Computação Paralela e Distribuída (CPD) do curso de Engenharia Informática. O objetivo deste trabalho é praticar o uso do *Dtrace*, numa máquina *Soláris 11*, para isso foram propostos dois exercícios, sendo que os seus resultados são apresentados ao longo deste relatório.**

1. Introdução

Como foi dito anteriormente, foram propostos dois exercícios para a praticar o uso do *Dtrace*. O primeiro exercício consiste em desenvolver uma script em *D* que faça um traçado das chamadas ao sistema *open()* (no caso de de uma máquina *Soláris 11* openat()), imprimindo por linha o nome do ficheiro executável, PID do processo, UID do utilizador e GID do grupo, o caminho absoluto para o ficheiro que for aberto, a cadeia de caracteres com as *flags* da chamada ao sistema openat() (O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_CREAT) e por fim o valor de retorno de chamada ao sistema. Este exercício contem um parte opcional, que consiste em modificar a script para sejam apenas detetados os ficheiros com *etc/* no caminho.

O segundo exercício proposto consiste, para todos os processos que estão no sistema, em contar o número de tentativas de abrir ficheiros existentes, o número de tentativas para criar ficheiros e contar o número de tentativas bem-sucedidas. Posteriormente a script deve imprimir, com um período (especificado em segundos) passado como argumento na linha de comandos, a hora e dia atual em formato legível e as estatísticas recolhidas por PID e respetivo nome.

2. Dtrace

O *Dtrace* é uma *Framework* para fazer traçados dinâmicos, esta *Framework* é usada para solucionar problemas no *Kernel* e aplicações em produção, em tempo real.

O *Dtrace* pode ser utilizado para se obter uma visão geral da execução do sistema, como a quantidade de memória, o tempo de CPU, os recursos usados por os processos activos. Esta *Framework* permite fazer traçados muito mais rebuscados e detalhados, tais como, por exemplo a lista de processos que tenta aceder a um ficheiro.

As administradores de sistemas escrevem programas em *D*, ajustando esses programas à informação que querem obter. A linguagem *D*, em termos de estrutura é muito semelhante ao *Awk*. Estes programas em *D*, consistem numa lista de uma ou mais provas e a cada prova está associada uma acção.

3. Exercicio 1

Para este exercício, desenvolvi uma script em *D*, script essa que faz um traçado das chamadas ao sistema openat() e imprime a seguinte informação por linha:

- Nome do ficheiro executável;
- PID do Processo;
- UID do Utilizador;
- GID do Grupo;
- Caminho absoluto para o ficheiro que for aberto;
- A cadeia de caracteres com as *flags* da chamada ao sistema openat() (O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_CREAT);
- O valor de retorno da chamada ao sistema.

Para alem de cada um destes tópicos, foi-nos proposto como exercício opcional, modificar a script de forma a só ser detetados os ficheiros com */etc* no caminho.

3.1. Script

A *Script* por mim criada contêm 2 provas, a primeira deteta à entrada a chamada ao sistema e guarda o *arg1* (que contem o caminho) na variável *self->path* bem como o *arg2* (que contem a *flag*) na variável *self->flags*. Na segunda prova no inicio, contem o predicado */strstr(self->path, "/etc") != NULL/* que permite apenas detectar os ficheiros com */etc* no caminho. Posteriormente nesta prova faço *strjoin* à variável *this->flags_string* da *string O_WRONLY* caso a condição *self->flags & O_WRONLY* se verifique, caso contrário verifica se a condição *self->flags & O_RDWR* é verdadeira e faz *strjoin* da *string O_RDWR*, caso a condição seja falsa faz *strjoin* da *string O_RDONLY*. Ainda nesta acção faço *strjoin* da *string |O_APPEND* caso a condição *self->flags & O_APPEND* se verifique, caso contrário faço *strjoin* da *string* , o mesmo se acontece para a *flag O_CREAT*.

No fim é imprimido no ecrã o nome do executável o PID do processo, o UID do utilizador, o GID do grupo, o caminho absoluto para o ficheiro que for aberto, a string com o conteúdo da variável *this->flag_string* e por fim o valor de retorno da chamada ao sistema.

```
/*
inline int O_RDONLY = 0;
inline int O_WRONLY = 1;
inline int O_RDWR = 2;
inline int O_APPEND = 8;
inline int O_CREAT = 256;
*/

this string flag_string;

syscall::openat:entry
{
        self->path = copyinstr(arg1);
        self->flags = arg2;
}

syscall::openat:return
/strstr(self->path, "/etc") != NULL/
{
    this->flags_string = strjoin(
        self->flags & O_WRONLY
        ? "O_WRONLY"
        : self->flags & O_RDWR
            ? "O_RDWR"
            : "O_RDONLY",
        strjoin(
            self->flags & O_APPEND  ? "|O_APPEND" : "",
            self->flags & O_CREAT   ? "|O_CREAT" : ""));

    printf("Executável: %s,%d,%d,%d, \"%s\",%s,%d\n", execname, pid, uid, gid
        ,self->path, this->flags_string, arg1);
}
```

3.2. Resultados Obtidos

Para testar a script executei alguns comandos de teste, propostos no enunciado, sendo que o resultado de cada comando pode ser consultado nas subsecções em baixo.

- Comando *cat /etc/inittab > /tmp/test* - Como podemos verificar pela tabela da figura 1, ao executarmos o comando *cat /etc/inittab > /tmp/test* na consola, este é logo detetado pelo *Dtrace* (1ª linha da tabela). Como ao executarmos o comando reencaminhados o *Output* para um novo ficheiro, na coluna das *flags* aparece a *flag* O_WRONLY que corresponde a leitura do ficheiro *inittab* e a *flag O_CREAT* que corresponde a criação do ficheiro *test_sergio*.

Execname	PID	UID	GID	Path	Flags	Valor de Retorno
bash	22177	29231	5000	/tmp/test_sergio	O_WRONLY|O_CREAT	4
cat	22177	29231	5000	/var/ld/ld.config	O_RDONLY	-1
cat	22177	29231	5000	/lib/libc.so.1	O_RDONLY	3
cat	22177	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
cat	22177	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
nfsmapid	1351	1	12	/etc/resolv.conf	O_RDONLY	10

Figura 1. Resultado Comando *cat /etc/inittab > /tmp/test*

- Comando cat /etc/inittab ≫ /tmp/test - Ao executarmos este comando estamos a fazer uma leitura do ficheiro *inittab*. Além disso este comando acrescenta o *Output* ao ficheiro *test_sergio* já existente, como tal é de esperar que na coluna referente às *flags* apareçam as *flags O_WRONLY, O_APPEND* e a *flag O_CREAT*, o que se verifica pela análise da 1ª linha da tabela da figura 2.

Execname	PID	UID	GID	Path	Flags	Valor de Retorno
bash	22182	29231	5000	/tmp/test_sergio	O_WRONLY|O_APPEND|O_CREAT	
cat	22182	29231	5000	/var/ld/ld.config	O_RDONLY	-1
cat	22182	29231	5000	/lib/libc.so.1	O_RDONLY	3
cat	22182	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
cat	22182	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1

Figura 2. Resultado Comando cat /etc/inittab ≫ /tmp/test

- Comando cat /etc/inittab | tee /tmp/test - Ao executarmos este comando a saída do comando *cat /etc/inittab* vai ser gravada no ficheiro *test* ao mesmo tempo em que é exibida no ecrá. O resultado obtido pela execução deste comando pode ser consultado na tabela da figura 3.

Execname	PID	UID	GID	Path	Flags	Valor de Retorno
cat	22187	29231	5000	/var/ld/ld.config	O_RDONLY	-1
cat	22187	29231	5000	/lib/libc.so.1	O_RDONLY	3
cat	22187	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
cat	22187	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
tee	22188	29231	5000	/var/ld/ld.config	O_RDONLY	-1
tee	22188	29231	5000	/lib/libc.so.1	O_RDONLY	3
tee	22188	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
tee	22188	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3
tee	22188	29231	5000	/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/SUNW_OST_OSLIB.mo	O_RDONLY	-1
nfsmapid	1351	1	12	/etc/resolv.conf	O_RDONLY	10

Figura 3. Resultado Comando cat /etc/inittab | tee /tmp/test

- Comando cat /etc/inittab | tee -a /tmp/test - Esfe comando faz exactamente o que faz o comando do tópico anterior, com a diferença que com a *flag -a* este acrescenta a saída do comando *cat /etc/inittab* ao ficheiro *test*. O resultado obtido pela execução deste comando pode ser consultado na tabela da figura 4.

Execname	PID	UID	GID	Path	Flags	Valo de Retorno
tee	22197	29231	5000	/var/ld/ld.config	O_RDONLY	-1
tee	22197	29231	5000	/lib/libc.so.1	O_RDONLY	3
tee	22197	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
tee	22197	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3
tee	22197	29231	5000	/usr/lib/locale/en_US.UTF-8/LC_MESSAGES/SUNW_OST_OSLIB.mo	O_RDONLY	-1
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
nfsmapid	1351	1	12	/system/volatile/rpc_door/rpc_100172.1	O_RDONLY	-1
cat	22196	29231	5000	/var/ld/ld.config	O_RDONLY	-1
cat	22196	29231	5000	/lib/libc.so.1	O_RDONLY	3
cat	22196	29231	5000	/usr/lib/locale/en_US.UTF-8/en_US.UTF-8.so.3	O_RDONLY	3
cat	22196	29231	5000	/usr/lib/locale/en_US.UTF-8/methods_unicode.so.3	O_RDONLY	3

Figura 4. Resultado Comando cat /etc/inittab | tee -a /tmp/test

4. Exercicio 2

Neste exercício, foi-nos proposto criar uma script que para cada processo em execução no sistema mostra-se as seguintes estatísticas:

- Número de tentativas de abrir ficheiros existentes;
- Número de tentativas para criar ficheiros;
- Número de tentativas bem-sucedidas.

Estas estatísticas devem ser imprimidas no ecrã, juntamente com a hora e dia, repetidamente, com um período (em segundos) passado como argumento na linha de comandos.

4.1. Script

Esta *Script* é constituida por 3 provas, primeira vai contar o número de tentativas para criar ficheiros, para isso esta prova contém um predicado que testa se a *flag* da chamada ao sistema *openat* é a *flag O_CREAT*, caso se verifique este predicado é incrementado um contador através de uma agregação com um *array* associativo (*@create[execname,pid]*). As outras duas provas, uma conta o numero de tentativas para abrir um ficheiro e a outra o numero de tentativas bem sucedidas, as ações associadas a estas provas processam-se da mesma forma que foi explicado atrás. No final é imprimido no ecrã o dia e a hora, o nome do executável, o PID, o contador do *array* *@create*, o contador do *array* *@open* e o contador do *array* *@success*, com um período de 2 segundos

```
/*
inline int O_RDONLY = 0;
inline int O_WRONLY = 1;
inline int O_RDWR = 2;
inline int O_APPEND = 8;
inline int O_CREAT = 256;
*/

syscall::openat:entry
/(arg2 & O_CREAT)==O_CREAT/
{
        @create[execname,pid] = count();
}

syscall::openat*:entry
/(arg2 & O_CREAT) == 0/ {
         @open[execname, pid] = count();
}

syscall::openat*:return
/ arg1 > 0 / {
        @success[execname, pid] = count();
}

tick-$1s
{

         printf ("%Y\n",walltimestamp);
         printf("%12s %6s %6s %6s %s\n","EXECNAME", "PID", "CREATE", "OPEN", "↩
               SUCCESS");
         printa ("%12s %6d %@6d %@6d %@d\n", @create, @open, @success);
         trunc(@create);
         trunc(@open);
         trunc(@success);
}
```

4.2. Resultados Obtidos

Os resultados obtidos na execução da *script*, referente ao exercício 2, podem ser consultados na figura 5. Estes resultados foram obtidos através da execução do seguinte comando:

```
dtrace -qs exercicio2.d 2
```

Sendo que 2, é o período (segundos) pelo qual os resultados são imprimidos no ecrã. De referir que nem sempre as provas detectam alguma coisa, pode haver períodos em que não há nenhuma atividade que satisfaça as provas definidas.

```
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:20
        sshd  23318      0      5 6
        bash  23320      0      6 3
        nscd   1447      0     12 12
        sshd  23319      0     20 19
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:22
      dtrace  23375      0      2 2
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:24
       utmpd    259      0      1 2
          ls  23376      0      5 4
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:26
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:28
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:30
        bash  23334      0      1 1
    nfsmapid   1351      0      2 0
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:32
  automountd   1443      0      1 1
        bash  23334      0      1 1
    nfsmapid   1351      0      3 1
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:34
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:36
        bash  23334      0      1 1
    nfsmapid   1351      0      4 0
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:38
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:40
        bash  23334      0      1 1
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:42
        bash  23334      0      1 1
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:44
          ls  23377      0      5 4
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:46
        bash  23334      0      1 1
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:48
        bash  23334      0      2 2
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:50
          rm  23378      0      4 3
    EXECNAME    PID CREATE   OPEN SUCCESS
2016 Apr 10 17:45:52
```

Figura 5. Resultado da *Script* do Exercicio 2

5. Conclusão

Como podemos verificar, ao longo deste trabalho, tivemos desenvolver duas scripts que nos permiti-se traçar/detectar o que nos foi pedido no enunciado. Ao longo do desenvolvimento apercebi-me cada vez mais das capacidades da *Framework Dtrace*, sendo que esta ferramenta nos permite, monitorizar ao pormenor tudo o que acontece num sistema. Para além desta grande vantagem, esta ferramenta também apresenta uma linguagem própria, a linguagem *D*, que como já foi referido anteriormente, é uma linguagem muito parecida (estruturalmente) com o *awk*.

Com esta linguagem, a *Framework* torna-se ainda muito mais ”poderosa”, uma vez que nos permite aproximar até ao máximo detalhe daquilo que pretendemos analisar no sistema.

Como trabalho futuro, pretendo aperfeiçoar os meus conhecimentos em *Dtrace*, porque, como já disse, para além de ser uma ferramenta poderosa, no que toca a administração de sistemas, também é uma ferramenta muito útil e acessível uma vez que a sua sintaxe não é muito difícil nem a sua estrutura, o que se torna difícil no estudo desta ferramenta é a numerosa informação que se dispõem, bem como as numerosas provas que é permitido se criar com o *Dtrace*.