

Log Rotate

Versão 1.0



HISTÓRICO DE VERSÕES

<i>Versão</i>	<i>Data</i>	<i>Autor da Versão</i>	<i>Alterações da Versão</i>
1.0	24/09/2020	Pedro Akira Danno Lima	Documento original Log Rotate .

RESUMO DO DOCUMENTO

<i>Descrição:</i>	Este documento descreve os processos de instalação, configuração e administração Log Rotate.
<i>Local de Publicação:</i>	
<i>Validade da Versão:</i>	20/11/2020
<i>Baseado no Modelo de Publicação Versão 1.0</i>	

Sumário



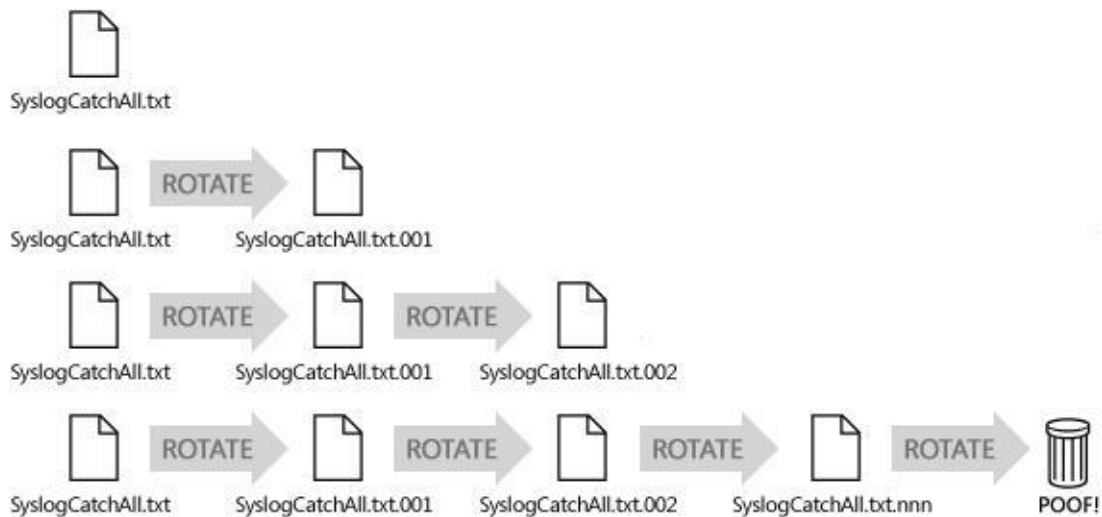
Log Rotate

Log file rotation

When you add an action to log messages to a file in Kiwi Syslog Server, you can choose to automatically rotate log files. Use log file rotation to prevent log files from growing indefinitely and using large amounts of disk space.

When log files are rotated:

- Messages are logged to the current log file. For example, SyslogCatchAll.txt.
- When the current log file reaches the specified size or age, it is renamed. For example, SyslogCatchAll.txt becomes SyslogCatchAll.txt.001. The logging process then creates a new, empty file with the original file name.
- When the new file reaches the specified size or age, the process is repeated. For example, SyslogCatchAll.txt.001 becomes SyslogCatchAll.txt.002, and SyslogCatchAll.txt becomes SyslogCatchAll.txt.001.
- When the maximum number of log files in the rotation have been created, the oldest is deleted.



Log Rotate

Log file rotation

To automatically rotate your log files:

1. From the Kiwi Syslog Service Manager, go to File > Setup.
2. Under the [rule](#), select the [Log to file action](#).
3. Select Enable Log File Rotation.
4. Specify the total number of log files in the rotation set.
5. Specify the rotation criteria:
 - To rotate files based on size, select Maximum log file size.
 - To rotate files based on age, select Maximum log file age.
6. Click OK.

Ref:

https://documentation.solarwinds.com/en/Success_Center/KSS/Content/KSS_AdminGuide_log_file_rotation.htm



Log Rotate

Log file rotation

Name of file:

file.log.1

file.log.2

file.log.3

file.log.4

file.log.nn

Name of **archive log file**:

file.arch.1

file.arch.2

file.arch.3

file.arch.4

file.arch.5

file.arch.nn



Log Rotate

Gcc or clang and GDB debugger

GDB debugger is command line debugger

```
[root localhost ~]# vim logrotate.c
```

```
[root localhost ~]# gcc -o exelogrotate logrotate.c
```

```
[root localhost ~]# gcc -Wall -O2 logrotate.c -o logrotate
```

```
[root localhost ~]# sudo apt-get install gdb
```

```
[root localhost ~]# gdb exelogrotate  
(gdb)
```

```
(gdb) run
```

```
(gdb) exit
```



Log Rotate

GDB debugger

O que é o GDB

O GDB (*GNU Project Debugger*) é uma ferramenta para:

- observar um programa enquanto este executa
- ver o estado no momento que a execução falha
- Permite:
 - iniciar a execução de um programa
 - executar linha-a-linha
 - especificar pontos de paragem
 - imprimir valores de variáveis
- Suporta C, C++, Objective-C, Ada e Pascal (entre outras linguagens)

Usar o GDB

- O GDB opera sobre *ficheiros executáveis* (não diretamente sobre o código-fonte)
- Para usar o GDB com um programa em C devemos compilar com opção **-g**:

```
[root localhost ~]# gcc -g -o programa programa.c
```

- A opção **-g** indica ao compilador para incluir no executável informação extra para o GDB

Log Rotate

GDB debugger

Usar o GDB

- Em seguida executamos gdb sobre o ficheiro executável compilado:

```
[root localhost ~]# gdb programa
```

- Após algumas mensagens obtemos a indicação de que o GDB está à espera de um comando:

```
...
```

```
(gdb)
```

Usar o GDB

- Tal como a *shell* de Linux, o GDB é um *programa interativo*:
 - lê um comando do teclado (até *Enter*)
 - processa o comando e mostra resultados
 - volta a esperar um novo comando
- O comando quit termina a sessão
 - alternativa: Ctrl-D (end-of-file)
 - podemos abreviar comandos
(e.q., escrever q em vez de quit)

Log Rotate

GDB debugger

Exemplo

- Vamos usar o GDB sobre um programa de exemplo
- O programa deveria calcular o *factorial* de um inteiro positivo

$n! = 1 \times 2 \times 3 \times \dots \times n$

- Exemplo: $4! = 1 \times 2 \times 3 \times 4 = 24$
- No entanto, o programa dá resultados errados

```
$ gcc -o factorial factorial.c
```

```
$ ./factorial
```

```
Introduza um inteiro positivo:4
```

```
Factorial 4 = 0
```

Programa errado

```
#include <stdio.h>
```

```
int main(void) {
```

```
    int n, i, fact;
```

```
    printf("Introduza um inteiro positivo:");
```

```
    scanf("%d", &n);
```

```
    for(i = 1; i <= n; i++)
```

```
        fact = fact*i;
```

```
    printf("Factorial %d = %d\n", n, fact);
```

```
}
```

Log Rotate

GDB debugger

Usar o GDB

- Vamos usar o GDB para perceber o erro
- Vídeo no YouTube:
[Introduction to GDB: a tutorial](#) (Harvard CS50)

Usando o GDB

Recompilar o programa com opção *debugging*:

```
[root localhost ~]# gcc -g -o factorial factorial.c
```

Executar o GDB com o executável compilado:

```
[root localhost ~]# gdb factorial
```

Executar o programa

Usamos o comando run para correr o nosso programa dentro do GDB:

```
(gdb) run
Starting program...
Introduza um inteiro positivo:4
Factorial 4 = 0
[Inferior 1 (process 9885) exited...
```

Log Rotate

GDB debugger

Executar o programa (cont.)

- O programa terminou e deu resultado errado
- Neste momento já não podemos observar o estado de variáveis
- Vamos correr novamente, mas desta vez pedir para parar a execução a meio
- Fazemos isso definindo um *breakpoint* no programa

Definir *breakpoints*

break *fun*

parar a execução no início da função *fun*

break *n*

parar a execução no início linha número *n*

Podemos usar *list* para listar o programa juntamente com números de linhas.

Observar valores de variáveis

print var

print *expr*

mostrar o valor de uma variável ou expressão

display var

display *expr*

mostrar o valor de uma variável ou expressão em cada *breakpoint*

Log Rotate

GDB debugger

Observar valores de variáveis

Vamos usar:

1. `break` para colocar um *breakpoint* no ciclo `for`;
2. `display` para mostrar os valores de `n`, `i`, e `fact`.

Conclusão

- Valor inicial de `fact` é incorreto
 - deveria ser 1 (porque $0! = 10! = 1$)
 - mas poderá ser 0 ou um valor arbitrário (dependendo da implementação)
- Isto ocorre porque **não inicializamos** a variável
- A correção é simples:

```
int n, i, fact = 1; // corrigido
```

Log Rotate

GDB debugger

Sumário de comandos

break *fun*

definir *breakpoint* no início da função *fun*

break *n*

definir *breakpoint* no início da linha *n*

delete *n*

remover o *breakpoint* número *n*

run

executar o programa desde o início

Sumário de comandos (cont.)

next

executar a próxima linha

step

executar a próxima linha (mas entra dentro de funções)

continue

continuar a execução até ao próximo *breakpoint* (ou até ao final)

Log Rotate

GDB debugger

Sumário de comandos (cont.)

print *expr*

calcular e mostrar o valor duma expressão

display *expr*

mostrar o valor da expressão sempre que parar

set var *nome* = *expr*

modificar o valor de uma variável

quit

terminar a sessão

Log Rotate

How Execute **logrotate.c**

```
[root localhost ~]# touch test1
```

```
#!/bin/bash
```

```
for (( ; ; ))
```

```
do
```

```
    echo "test"
```

```
    sleep 10
```

```
done
```

```
[root localhost ~]# chmod +x test1
```

```
[root localhost ~]# gcc -Wall -O2 logrotate.c -o logrotate
```

```
[root localhost ~]# touch file.log
```

```
[root localhost ~]# ./test1 | ./logrotate file1.txt 1000
```



Log Rotate

In **srv** directory

Logrotate.c -> simple logrotate that have the just 2 files. File.log and file.log.old

Logrotate2.c -> can choose how much file.logs you need. Ex: **3** file.log.1 file.log.2 file.log.3

Works with rotation

Logrotate3.c -> use a single process to logrotate and **archive**. In the same program. Logrotate.c

Logrotate4.c -> versão melhorada do logrotate3.c usando system call e buffer para o **archivelog** ao invés de fopen... etc

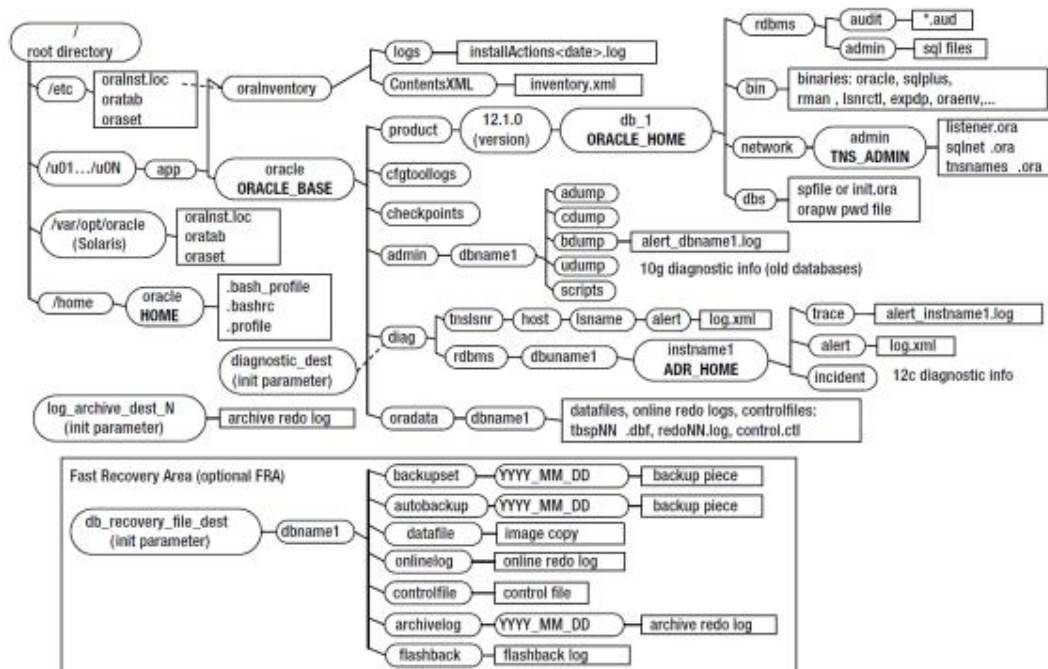
Logrotate.5.c -> IPC to communication with archivelog. Implementation archivelog

Arc.log -> archivelog



Log Rotate

Optimal Flexible Architecture Log (OFAL)



/u01/app/log/ <version>

/logdata

/file.log.nn

/arch

/log_archive_dest.arc.nn

/logspar[ammeter]

/spfile or innit.ora



FICHA TÉCNICA

Elaboração

Pedro Akira Danno Lima

Colaboração

Guilherme Augusto di Stefano

Revisão da Versão

Guilherme Augusto di Stefano

Versão 1.0

Setembro / 2020