Conceitos de Serviços
Agendador de tarefas cron
Serviço de impressão
System Log

Serviços cron, cups, rsyslog e syslog-ng

Setembro/2017

Prof. Jairo

jairo@uni9.pro.br

professor@jairo.pro.br

http://www.jairo.pro.br/

Este material tem por única intenção reunir um conteúdo acadêmico necessário para auxiliar no ensino da disciplina "Prática e Administração de Sistemas Operacionais de Redes Livres", ministrado nos cursos de Tecnologias em Redes de Computadores e Segurança da Informação.

O arquivo "conceitos-servicos.pdf" pode ser livremente acessado em "http://www.jairo.pro.br/prat_adm_sist_oper/".

Qualquer crítica ou sugestão, favor entrar em contato com o Prof. Jairo no endereço eletrônico "jairo@uni9.pro.br" ou "professor@jairo.pro.br".

Sumário

1 - CONCEITOS DE SERVIÇOS

Os serviços, assim como as demais aplicações, são sustentados por processos do sistema operacional.

O processo de serviço rodando num sistema operacional membro da família Unix é chamado de **DAEMON**, de **D**isk **A**nd **E**xecution **MON**itor. É devido a isso que, de um modo geral, os processos de serviço tem nomes que terminam com a letra **d**. Por exemplo, inetd, httpd, sshd, named.

Um DAEMON é um tipo de processo que roda em background, e uma vez iniciado não finaliza nunca. Seu objetivo é escutar e atender às requisições dos clientes do serviço.

Os DAEMONS podem ser de duas naturezas, **inetd** ou **standalone**. A diferença é que o DAEMON inetd pode suportar simultaneamente mais de um serviço, por exemplo, serviços FTP e TELNET. Já o DAEMON standalone suporta um único serviço.

No Linux, é comum usar **xinetd** ao invés de inetd. A diferença entre **inetd** e **xinetd** ocorre nos arquivos de configuração: enquanto inetd é configurado pelo arquivo **/etc/inetd.conf**, xinetd tem arquivos específicos de configuração no diretório **/etc/xinetd.d**.

Tipicamente, do ponto de vista do sistema operacional, um serviço é constituído de pelo menos três partes:

- o executável [que ao rodar dá origem ao processo DAEMON];
- o arquivo de configuração do serviço;
- o script (ou arquivo ***unit file***) de inicialização do serviço.

Por exemplo, no serviço SSH de um CentOS 6 [inicialização SystemV] temos o executável "**/usr/sbin/sshd**", o arquivo de configuração "**/etc/ssh/sshd_config**" e o script de inicialização "**/etc/init.d/sshd**". Num sistema SystemD (por exemplo, CentOS 7 ou Ubuntu 16), ao invés do script de inicialização existe o arquivo sshd.service, que é o ***unit file*** do serviço, em /lib/systemd/system.

Alguns daemons rodam como **serviço local**, outros como **serviço em rede**. A diferença é que o serviço local só atende a clientes na mesma máquina que o serviço, já o serviço em rede atende a clientes pela rede. O serviço em rede, para funcionar, necessita da infraestrutura de comunicação em rede.

Um exemplo de **serviço local** é o cron. Cron é o agendador de tarefas padrão dos sistemas da família Unix.

Um exemplo de **serviço em rede** é o serviço web Apache.

Uma rede de computadores pode ser definida como um conjunto de módulos processadores[1] interligados por um meio físico[2], e que fazem uso de um protocolo comum de comunicação. Por protocolo de comunicação entendemos um conjunto de regras de comunicação comum às partes envolvidas nessa troca de dados.

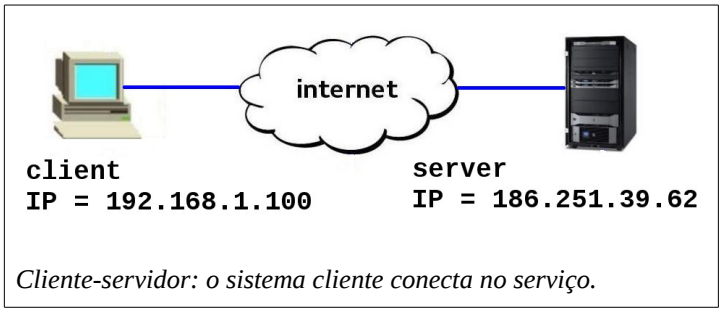
Na nossa proposta de trabalhar em ambientes livres, usamos apenas protocolos de comunicação abertos. Daí a escolha pela suíte[3] Internet Protocol [TCP/IP].

Atualmente, a suíte TCP/IP é o protocolo de comunicação *de fato*. De fato significa que não é o único protocolo de comunicação em rede existente, porém é o mais praticado em qualquer realidade de rede de computadores.

O protocolo IP faz uso de endereços IPs, que podem ser de versão 4 (ipv4) ou 6 (ipv6).

Na versão 4, o endereço IP tem 4 bytes [32 bits], que possibilita apenas [teoricamente] 2^{32} endereços [4294967296]. Devido a isso, atualmente já se verifica a exaustão dos endereços na versão 4, pela expansão da base de usuários na internet. Consequentemente, está iniciando a adoção em massa da versão 6, que permite teoricamente 2^{128} endereços. Porém, nas redes internas não existe esta falta de endereços IP na versão 4, então atualmente a adoção do ipv6 está ocorrendo apenas na internet.

Server ou host significa hospedador, ou máquina servidora que hospeda o serviço. Para acessar determinado serviço usa-se uma aplicação cliente, daí o conceito de cliente-servidor. Do ponto de vista do sistema operacional, as aplicações servidora e cliente não passam de aplicações comuns, com a diferença de normalmente precisarem prover acesso em rede. Isso implica em processos do sistema operacional tanto do lado do cliente quanto processos do lado servidor, tendo a rede como veículo a transportar as informações de um para o outro.

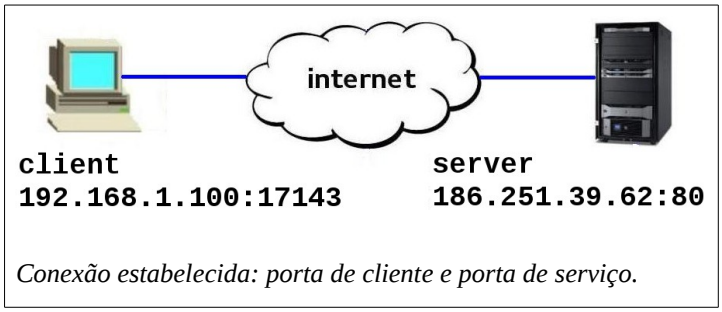


Cliente-servidor: o sistema cliente conecta no serviço.

O endereçamento IP é hierárquico: para o cliente alcançar determinado serviço, precisa primeiro acessar a rede do host (server), chegando a essa rede alcança o host e, nesse host, busca pela porta do serviço.

Exemplo: 186.251.39.62:80.

No exemplo acima, 186.251.39.0/24 é o endereço da rede [nesse caso, classe C] do host,



Conexão estabelecida: porta de cliente e porta de serviço.

1 Por exemplo: computadores, roteadores, switches, hubs.
2 Cabo de rede, cabo coaxial, cabo ótico, wireless.
3 Suíte nos sentido de conjunto de protocolos divididos em camadas funcionais, num conceito de arquitetura de rede.

186.251.39.62 é o endereço deste host nesta rede, e 80 é a porta do serviço hospedado por este host [80 é a porta padrão do serviço web, que usa protocolo de camada de aplicação HTTP].

Devido às portas de serviço é possível, por exemplo, hospedar dois serviços web no mesmo host [mesmo endereço IP], desde que não usem a mesma porta TCP.

Pelo seu lado, para acessar o serviço, a aplicação cliente precisa apenas saber o endereço desse host, já que usa o mesmo protocolo de camada de aplicação do serviço e conhece a porta padrão [default] deste serviço. Se o serviço estiver em outra porta que não a default, esta informação precisa ser passada ao cliente ou ele não conseguirá acesso.

Um vez iniciado determinado serviço num host, ele abre a porta deste serviço e a mantém aberta enquanto estiver rodando. Desse modo, aguarda pelo acesso dos clientes. Já o cliente, no acesso a este serviço, abre uma porta de cliente para estabelecer a conexão e receber os dados enviados pelo servidor.

A faixa de **portas de serviço** normalmente estão entre 0 e 1023. Isso não implica em dizer que não possa haver serviço em portas fora dessa faixa. Por exemplo, o serviço proxy squid usa a porta padrão 3128.

Já as **portas de cliente** estão sempre acima de 1023.

Para os serviços em rede, normalmente são usados os protocolos de transporte TCP ou UDP.

Alguns exemplos de portas padrão **TCP**, associadas a protocolos de camada de aplicação, são:

PORTA (porta segura)	PROTOCOLO	NOME DO SERVIÇO
21	FTP	File Transfer Protocol
22	SSH	Secure Shell
23	TELNET	Telecommunications Network Protocol. Acesso remoto shell
25 (587)	SMTP (SMTPS)	Simple Mail Transfer Protocol
53	DNS	Domain Name System
80 (443)	HTTP (HTTPS)	Hyper Text Transfer Protocol
110 (995)	POP (POPS)	Post Office Protocol
143 (993)	IMAP (IMAPS)	Internet Message Access Protocol
3306	MYSQL	MySQL Database system

Alguns exemplos de portas padrão **UDP**, associadas a protocolos de camada de aplicação, são:

PORTA	PROTOCOLO	NOME DO SERVIÇO
67	DHCP (Server)	Dynamic Host Configuration Protocol
68	DHCP (Client)	Dynamic Host Configuration Protocol
53	DNS	Domain Name System

1.1 - Serviço local

O serviço é local quando o cliente só consegue acessar este serviço quando estiver na mesma máquina do serviço. Como exemplo, temos o serviço cron.

1.2 - Serviço em rede

O serviço é em rede quando o cliente acessa o serviço remoto pela rede. Neste caso é necessário que exista a infraestrutura de rede, endereços IP tanto no sistema que hospeda o serviço quanto no cliente, além de roteamento para que um acesse o outro.

2 - SERVIÇO CRON

2.1 - Conceitos básicos

O cron[4] é um serviço local. É o agendador de tarefas padrão nos sistemas da família Unix. Por exemplo, se for necessário executar determinada aplicação às 03:00, não é necessário o usuário acessar o servidor neste momento, pois ele pode deixar esta execução agendada.

Este serviço é usado para disparar executáveis em datas previamente agendadas, a partir da configuração de uma tabela chamada **crontab**. Esta tabela especifica o que deve ser executado em qual mês, semana, dia, hora e minuto.

A crontab é uma tabela individual e única para cada usuário. Esta tabela é um arquivo de texto, onde cada atividade agendada está numa nova linha do arquivo.

4 A palavra cron vem de chronos, o deus do tempo da mitologia grega.

Para saber se o serviço local cron está rodando, num Ubuntu SystemV (anterior à versão 14), usar o script de administração do serviço, **/etc/init.d/cron**:

```
root# /etc/init.d/cron  status
…
cron start/running, process 985
```

NOTA 1: Num Linux, poderia ser usado o comando "**service cron status**" (ou service crond status).

NOTA 2: Num Linux SystemD, como por exemplo um Red Hat 7, o comando seria "**systemctl status crond**".

Como o serviço é sustentado por um ou mais processos, também poderia ser procurado pelo processo do serviço:

```
root# ps  -ef  |  grep  cron
root      985    1   0  21:06 ?        00:00:00      cron
```

Se fosse necessário iniciar o serviço, bastaria passar o parâmetro **start** ao invés de status. Por exemplo:

```
root# /etc/init.d/cron  start
```

NOTA: analogamente, se fosse necessário parar o serviço, bastaria passar o parâmetro **stop** ao invés de status.

Se fosse necessário incluir o serviço **cron** na inicialização do sistema, num Ubuntu SystemV o comando seria:

```
root# update-rc.d  cron  defaults
Adding system startup for /etc/init.d/cron ...
   /etc/rc0.d/K20cron -> ../init.d/cron
   /etc/rc1.d/K20cron -> ../init.d/cron
   /etc/rc6.d/K20cron -> ../init.d/cron
   /etc/rc2.d/S20cron -> ../init.d/cron
   /etc/rc3.d/S20cron -> ../init.d/cron
   /etc/rc4.d/S20cron -> ../init.d/cron
   /etc/rc5.d/S20cron -> ../init.d/cron
```

NOTA: Num Red Hat SystemV o comando é **chkconfig crond on**. Nos sistemas operacionais que usam a inicialização SystemD, o comando é **systemctl enable crond** (ou systemctl enable cron).

2.2 - Crontab

Uma vez confirmado que o serviço **cron** está rodando, as tarefas agendadas serão iniciadas no horário configurado na **crontab**.

O comando **crontab** é usado tanto para listar o conteúdo da tabela cron [caso exista], quanto para incluir uma nova entrada ou modificar uma entrada existente.

```
shell$ id
uid=1000(aluno) gid=1000(aluno) grupos=1000(aluno),3(sys)
shell$ crontab  -l
no crontab for aluno
```

No comando acima, a opção **-l** (letra *ele*)é para listar o conteúdo da crontab. Neste caso, o usuário aluno não tem nenhuma tarefa agendada na crontab.

A seguir, o usuário aluno irá incluir uma entrada na crontab para disparar o comando **/bin/date** a cada 2 minutos.

Antes, por garantia, deve alterar a variável de ambiente **EDITOR**, para definir o editor **vi** como o *default* nesta edição:

```
shell$ export  EDITOR=vi
shell$ echo  $EDITOR
vi
```

Agora, então, editar a crontab com a opção **-e** no comando crontab:

```
shell$ crontab  -e
no crontab for aluno - using an empty one
```

```
*/2  *  *  *  *  /bin/date  >>  /tmp/date.txt  2>&1
~
~
~
-- INSERT --
```

O comando "crontab -e" acima, invocou o editor padrão, que no caso é o vi. No modo INSERÇÃO, foi incluída a linha que agendou a execução da tarefa.

Na crontab, os cinco campos de agendamento de horário, pela ordem, são:

1º - minutos [0-59];
2º - horas [0-23];
3º - dia do mês [1-31];
4º - mês [1-12];
5º - dia da semana [0-6, onde 0 é domingo e 6 sábado].

Se algum campo permanecer com o símbolo * [asterisco], será executado todas as vezes. Por exemplo, se for no campo da horas [2º], vai executar a toda hora.

Ainda no exemplo acima, no campo de minutos foi usado ***/2**, pois originariamente o menor intervalo de tempo possível era 5 minutos, então com a construção acima é possível disparar a execução em intervalos menores do que 5 minutos.

Para verificar a tarefa agendada, comandar de novo "**crontab -l**":

```
shell$ crontab  -l
*/2  *  *  *  *  /bin/date  >>  /tmp/date.txt  2>&1
```

Depois de alguns minutos, verificar se a tarefa agendada está mesmo rodando a cada 2 minutos. Isto pode ser verificado visualizando o conteúdo do arquivo **/tmp/date.txt**:

```
shell$ more  /tmp/date.txt
Sat Jul 15 21:46:01 -03 2017
Sat Jul 15 21:48:01 -03 2017
Sat Jul 15 21:50:01 -03 2017
Sat Jul 15 21:52:01 -03 2017
```

Abaixo segue alguns exemplos de agendamentos na crontab:

0	*****	*****	*****	*****	=> executa a toda hora cheia;
15	**2**	*****	*****	*****	=> diariamente, às 02:15;
20	**11**	**10**	*****	*****	=> dia 10 de todo mês, às 11:20;
30	**23**	*****	**6**	*****	=> apenas no mês de junho, todos os dias às 23:30;
25	**4**	*****	*****	**0**	=> apenas aos domingos, às 04:25;
10	**3**	**15,30**	**5**	*****	=> apenas no mês de maio, nos dias 15 e 30 às 03:10.

NOTA: a crontab não verifica a validade da data inserida. Por exemplo, se for agendada uma atividade para o dia 30 de fevereiro, esta nunca será executada.

3 - SERVIÇO CUPS

O serviço de impressão default nas distribuições Linux é o **cups**. CUPS são as iniciais de **C**ommon **U**nix **P**rinting **S**ystem.

O cups é um serviço que pode atender ao cliente em rede, mas que normalmente vem configurado para permitir somente acesso local, então na maior parte dos casos pode ser considerado também como serviço local.

Serviço local cups não significa que não possa usar impressoras na rede, significa apenas que o cliente deste serviço precisa ser local, isto é, precisa estar na mesma máquina aonde roda o serviço cups.

Num Ubuntu, para saber se o pacote cups está instalado, comandar:

```
shell$ dpkg  -l | grep  -i  cups
```

NOTA 1: num Red Hat (ou CentOS) o comando seria "**rpm -aq | grep -i cups**".

NOTA 2: se fosse necessário instalar o cups num Ubuntu, o comando seria "**apt-get install cups**", e num Red Hat (ou CentOS) este comando seria "**yum install cups**".

Num servidor de impressão, o serviço cups precisa estar rodando. Num Linux com inicialização SystemV, para saber se o serviço está rodando, comandar:

```
shell$ /etc/init.d/cups  status
cups start/running, process 505
```

NOTA 1: se o sistema fosse SystemD, o comando seria "**systemctl status cups**".

NOTA 2: se fosse o caso de precisar iniciar o serviço, num sistema SystemD precisaria comandar "**/etc/init.d/cups start**", e num SystemD precisaria comandar "**systemctl start cups**".

As configuracões das impressoras estão no arquivo **/etc/cups/printers.conf**, e o queue[5] de jobs em **/var/spool/cups**.

Para adicionar uma impressora, usar o comando **lpadmin**:

```
root# lpadmin -p printer02 -v socket://192.168.1.101:9100 -E
```

No comando acima foram usadas as seguintes opções:

-p: o nome da impressora;
-v: dispositivo (device), que suporta impressão diretamente usando TCP/IP, também chamado de AppSocket ou JetDirect. Neste caso trata-se de uma impressora de rede, que atende no endereço IP 192.168.1.101 e porta 9100;
-E: para habilitar o destino e passar a aceitar jobs de impressão nesta fila.

Para saber o status das impressoras, usar o comando **lpstat**:

```
root# lpstat -v
device for printer02: socket://192.168.1.101:9100
```

No comando acima, a opção "**-v**" foi para imprimir uma lista com o status de todas as impressoras.

NOTA: se fosse usado também a opção "**-t**", mostraria todas as informações de status das impressoras.

5 Queue: queue é uma sequência de tarefas que aguardam serem processadas. No caso da impressora, trata-se de uma fila de jobs aguardando serem impressos.

Para mostrar o status de uma impressora específica, comandar:

```
root# lpstat -p printer02
printer printer02 is idle.  enabled since Sat Jul 15  17:12:48  2017
```

Para mostrar ou reconfigurar as opções e defaults da impressora, usar o comando **lpoptions**:

```
root# lpoptions -p printer02
auth-info-required=none copies=1 device-uri=socket://192.168.1.101:9100 finishings=3 job-hold-
until=no-hold job-priority=50 job-sheets=none,none marker-change-time=0 number-up=1 printer-
info=printer02 printer-is-accepting-jobs=true printer-is-shared=true printer-location printer-make-and-
model='Local Raw Printer' printer-state=3 printer-state-change-time=1505679168 printer-state-
reasons=none printer-type=4 printer-uri-supported=ipp://localhost:631/printers/printer02
```

Para alterar o tamanho do papel para A4, comandar:

```
root# lpoptions -p printer02 -o PageSize=A4
auth-info-required=none copies=1 device-uri=socket://192.168.1.101:9100 finishings=3 job-hold-
until=no-hold job-priority=50 job-sheets=none,none marker-change-time=0 number-up=1 PageSize=A4
printer-info=printer02 printer-is-accepting-jobs=true printer-is-shared=true printer-location printer-
make-and-model='Local Raw Printer' printer-state=3 printer-state-change-time=1505679168 printer-
state-reasons=none printer-type=4 printer-uri-supported=ipp://localhost:631/printers/printer02
```

NOTA: as opções vão para o arquivo de configuração **/etc/cups/lpoptions**.

Para o usuário verificar o status da impressora printer02, usar o comando **lpq**:

```
shell$ lpq -P printer02
printer02 is ready
no entries
```

Para o usuário imprimir o arquivo arq.txt na impressora printer02, o comando é **lpr**:

```
shell$ lpr -P printer02 arq.txt
shell$ lpq -P printer02
printer02 is ready and printing
Rank    Owner          Job     File(s)         Total Size
active  (null)         1       untitled        1024 bytes
```

Para o usuário cancelar o queue de impressão, usar o comando **lprm**:

```
shell$ lprm -P printer02
shell$ lpq -P printer02
printer02 is ready
no entries
```

Para incluir uma impressora como a default, usar o comando **lpoptions**:

```
shell$ lpoptions -d printer02
```

Uma vez definida a impressora como a default, quando for acessá-la não será mais necessário usar a opção "**-P printer02**":

```
shell$ lpq
printer02 is ready
no entries
```

Outros comandos usados pelo root para administrar o cups, são:

lpadmin -x printer02 => remove a impressora

lpstat -o => lista todos os jobs de impressão

cancel -a => cancela todos os jobs de impressão

cupsdisable printer02 => desabilita a impressora, para não receber mais jobs de impressão dos usuários

cupsenable printer02 => retorna a impressora a receber jobs

3.1 - Serviço cups via interface web

Outra maneira de administrar o cups é via interface web, com um navegador. Porém, pelo default, deixa apenas o cliente local usar esta ferramenta. Por isso, será necessário reconfigurar o cups para deixar o root acessar remotamente a interface web.

Para isso, editar o arquivo cupsd.conf, mas primeiro guardar uma cópia de segurança:

```
root# cd  /etc/cups
root# cp  cupsd.conf  cupsd.conf.SAVE
root# vi  cupsd.conf
```

```
#Listen localhost:631
Port 631
```

Nas configurações acima, foi comentada a linha "Listen localhost:631" e incluída a linha "**Port 631**".

Além da modificação acima, e pensando na segurança, foi limitado o acesso ao serviço de impressão cups para apenas as máquinas da rede 192.168.0.0/16. Isto foi feito editando as diretivas "**<Location />**" e "**<Location /admin>**", também no arquivo de configuração **cupsd.conf**, conforme mostrado abaixo:

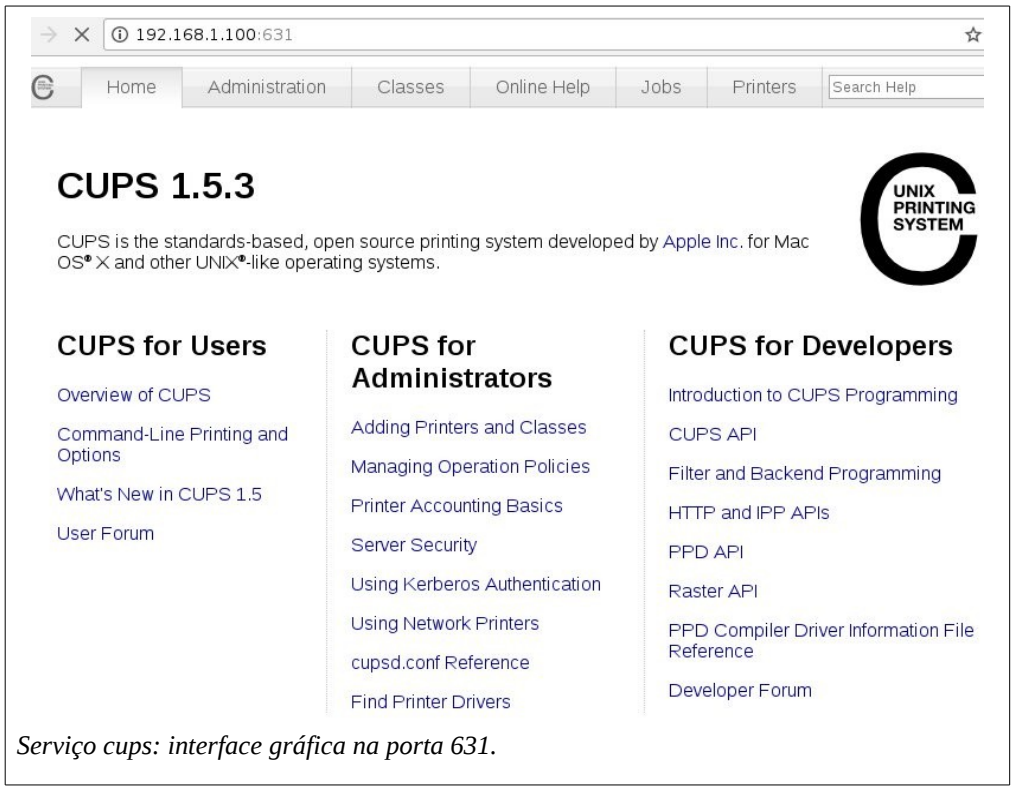
```
<Location />
#  Order allow,deny
   Order deny,allow
   Allow 192.168.0.0/16
</Location>

# Restrict access to the admin pages...
<Location /admin>
#  Order allow,deny
   Order deny,allow
   Allow 192.168.0.0/16
</Location>
```

Depois disso, o serviço cups precisa ser recarregado. Num sistema SystemV, é usado o comando abaixo:

```
root# /etc/init.d/cups  reload
```

NOTA: se fosse um sistema SystemD, o comando seria "**systemctl restart cups**".

Agora, basta abrir um navegador (por exemplo, Chrome) e apontar para a porta 631 do servidor cups, por exemplo, http://192.168.1.100:631, conforme figura abaixo:



Serviço cups: interface gráfica na porta 631.

NOTA: se for realizar alguma tarefa administrativa, tal como criar ou modificar uma impressora, será solicitado um usuário e senha. O default é enviar o usuário root.

3.2 - Cliente remoto de serviço cups

Os exemplos de impressão acima foram feitos no próprio servidor de impressão. Porém, não é necessário instalar e rodar o serviço cups para conseguir imprimir numa impressora remota.

Para um cliente conseguir acessar o serviço de impressão cups, precisa apenas instalar e configurar o pacote **cups-client**.

Num Ubuntu, o comando para instalar este pacote é:

```
client-root# apt-get  install  cups-client
```

NOTA: num Red Hat o comando seria "**yum install cups-client**".

Depois de instalado, precisa criar e configurar o arquivo **/etc/cups/client.conf**, de modo que aponte para o servidor de impressão:

```
client-root# vi  /etc/cups/client.conf
```

```
ServerName  192.168.1.100:631
~
~
-- INSERT --
```

NOTA: no arquivo de configuração do cliente de impressão, precisa declarar o nome ou endereço IP do serviço cups, além da porta do serviço.

Depois disso, é só testar o acesso à impressora:

```
client-root# lpq  -P  printer02
printer02 is ready
no entries
```

4 - SERVIÇO DE LOG

O serviço de log é usado pelas aplicações e serviços, para enviarem mensagens de eventos a um servidor de log. Este serviço registra em arquivos as mensagens de eventos.

Os eventos são alertas, informações, erros, notificações e avisos gerados pelo sistema operacional e pelas aplicações, e que têm por finalidade informar sobre a ocorrência de falhas,

defeitos, acessos, tentativas de acesso, etc.

4.1 - Arquivos de log

Para armazenar os logs são usados arquivos, que podem tanto ser locais ao sistema que os gerou, quanto remotos. Do ponto de vista da segurança, é melhor guardar os logs num servidor de logs remoto, pelos seguintes motivos:

i. o servidor de logs pode centralizar os arquivos logs de todos os demais servidores. Isso facilita tanto na análise e pesquisa de eventos quanto no backup;
ii. em caso de invasão ou exploração de vulnerabilidade, o evento logado num servidor remoto fica inacessível ao invasor, que portanto não terá como excluí-lo ou modificá-lo para apagar rastros.

Os arquivos de log contém informações sobre software, hardware, processos e componentes do sistema. Pela revisão dos dados armazenados nos logs, pode-se fazer uma pesquisa de problemas (troubleshooting) que sirva de diagnóstico para identificar a fonte destes problemas.

No default dos sistemas Linux, os arquivos de logs são guardados no diretório **/var/log**. Normalmente estes arquivos são de texto. Nos Unix, normalmente vão para o diretório /var/adm.

O principal arquivo de logs é o **/var/log/messages**. Mas para a distribuição Debian e derivados, este arquivo foi substituído pelo /var/log/syslog.

Por exemplo, para visualizar as últimas linhas do arquivo **/var/log/boot.log**, comandar:

```
root# tail  -5  /var/log/boot.log
 * Starting regular background program processing daemon                [ OK ]
 * Starting deferred execution scheduler                                [ OK ]
 * Starting CPU interrupts balancing daemon                             [ OK ]
 * Starting CUPS printing spooler/server                                [ OK ]
 * Starting crash report submission daemon                              [ OK ]
```

Embora a grande maioria dos arquivos de logs sejam de texto, isto não é uma regra. Por exemplo, o arquivo **/var/log/lastlog**, que é usado para registrar os logins de acessos, não é um arquivo de texto, e portanto não pode ser visualizado com os comandos more, tail, etc.

```
root# file  /var/log/lastlog
/var/log/lastlog: data
```

Deste modo, para visualizar os logins de acesso, usar o comando **last**. Este comando lê o database de logins no arquivo /var/log/lastlog:

```
root# last | tail
root     pts/0        192.168.0.12     Sat Jul 15 21:07 - down   (01:52)
reboot   system boot  3.13.0-32-generi Sat Jul 15 21:06 - 22:59  (01:53)
root     pts/1        192.168.0.12     Fri Jul 14 15:07 - down   (02:13)
aluno    pts/0        :0               Fri Jul 14 15:05 - down   (02:15)
reboot   system boot  3.13.0-32-generi Fri Jul 14 15:05 - 17:21  (02:16)
aluno    pts/1        :0               Fri Jul 14 15:00 - down   (00:03)
aluno    pts/0        :0               Fri Jul 14 14:56 - down   (00:07)
reboot   system boot  3.13.0-32-generi Fri Jul 14 14:56 - 15:04  (00:07)
```

Já o comando **dmesg** é usado para mostrar em tempo real as mensagens de buffer[6] do kernel (kernel ring buffer), mas que somente registra no arquivo /var/log/dmesg durante o processo de boot. Para filtrar a saída do comando dmesg e mostrar apenas linhas que contenham as palavras fail, error, warning, alert, emerg ou crit, usar a linha de comando abaixo:

```
root# dmesg | grep -ie 'fail\|error\|warning\|alert\|emerg\|crit'
[    0.024733] acpi PNP0A03:00: _OSC failed (AE_NOT_FOUND); disabling ASPM
[    0.024767] acpi PNP0A03:00: fail to add MMCONFIG information, can't access extended PCI
configuration space under this bridge.
[    1.718292] EXT4-fs (vda1): re-mounted. Opts: errors=remount-ro
[    2.985288] init: failsafe main process (729) killed by TERM signal
```

NOTA: o comando dmesg é usado nos sistemas SystemV. Nos sistemas SystemD, o comando equivalente é **journalctl**:

```
root# journalctl | grep -ie 'fail\|error\|warning\|alert\|emerg\|crit'
Jul 15 13:08:40 pinguim dbus[755]: [system] Failed to activate service 'org.bluez': timed out
Jul 15 13:08:40 pinguim dbus-daemon[755]: dbus[755]: [system] Failed to activate service 'org.bluez':
timed out
Jul 15 18:39:54 pinguim qemu-kvm[7470]: gssapiv2_client_plug_init() failed in
sasl_server_add_plugin(): generic failure
Jul 15 18:39:54 pinguim qemu-kvm[7470]: _sasl_plugin_load failed on sasl_server_plug_init for plugin:
gssapiv2
Jul 15 18:39:54 pinguim virtlogd[7438]: End of file while reading data: Input/output error
```

6 Buffer: buffer é uma porção da memória, mantida separada e usada temporariamente como um espaço para armezenar dados que estejam sendo enviados ou recebidos por um dispositivo externo, como um disco, teclado ou impressora.

4.2 - Logger

O **logger** é um comando que faz interface com o serviço de log, e é usado para enviar mensagens para os arquivos de logs. Esta característica do logger o torna muito útil tanto nos scripts shell quanto para diagnóstico do serviço de log.

Por exemplo, num Ubuntu12, o usuário aluno executa o seguinte comando:

```
shell$ logger  "teste de aluno"
```

Nesta máquina, o root verifica o final do arquivo /var/log/syslog:

```
root# tail  /var/log/syslog
...
Jul 15 16:22:37 ubuntu12 root: teste de aluno
```

A mensagem do logger pode ser direcionada para um arquivo de log previamente definido, por exemplo, /var/log/mail.log. Neste caso, o usuário precisa usar a opção **-p**:

```
shell$ logger  -p  mail.alert  "teste de log de email"
```

No comando acima, a opção **-p** significa prioridade (priority), e foi usada para registrar a mensagem na facilidade (facility) mail, com nível de alerta (alert). A prioridade foi especificada num par "facilidade.nível" (facility.level), no caso acima, **mail.alert**.

NOTA: o par "facilidade.nível" precisa estar previamente configurado no serviço de log.

Depois disso, o root pode verificar o log no arquivo **/var/log/mail.log**:

```
root# tail  /var/log/mail.log
...
Jul 15 17:05:55 ubuntu12 root: teste de log de email
```

NOTA: a configuração de um par "facilidade.nível" no serviço de log, envolve também a definição de um arquivo de log para guardar as mensagens desta facilidade.

4.3 - Serviço rsyslog

O serviço syslog (syslogd, rsyslog ou rsyslogd) é o serviço de registro de logs padrão dos sistemas da família Unix. Normalmente é instalado por padrão, afinal não dá para imaginar um sistema operacional que funcione sem serviço de logs.

Num Linux, o serviço é o **rsyslog**, cujo arquivo de configuração é **/etc/rsyslog.conf**. Uma inspecionada neste arquivo mostra as configurações das facilidades associadas aos respectivos arquivos de log:

```
root# grep -v "^#" /etc/rsyslog.conf | grep -v "^$"
...
kern.debug                                      /var/log/firewall.log
*.info;mail.none;authpriv.none;cron.none        /var/log/messages
authpriv.*                                      /var/log/secure
mail.*                                          -/var/log/maillog
cron.*                                          /var/log/cron
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

No comando acima, "^#" é uma expressão regular, e significa encontrar o caractere cerquilha como primeiro da linha. No segundo grep, "^$" é a expressão regular para linha em branco. Deste modo, a linha de comando acima vai dar como saída apenas as linhas que não iniciam por cerquilha e que não sejam linhas em branco.

NOTA: no lado esquerdo do arquivo de configuração ficam os pares "facilidade.nível", e no lado direito estão definidos os arquivos de log para onde encaminhar a mensagem que se enquadre na facilidade.

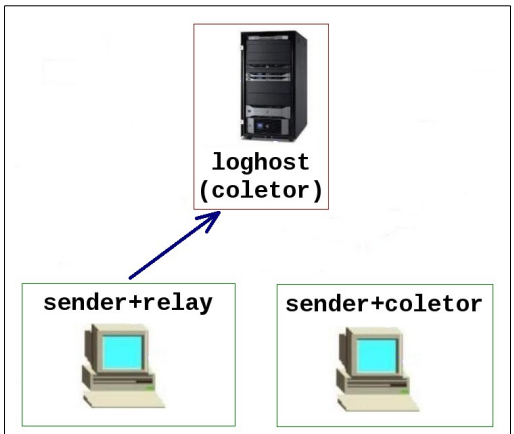
4.3.1 - Loghost

Loghost é nome genérico para o host que centraliza as funções de syslog. Neste caso, ao invés de cada sistema operacional logar as mensagens em arquivos locais, estas são enviadas para o loghost.

Para transmitir as mensagens através da rede, é usado o protocolo SYStem LOG, que pode tanto usar protocolo de transporte UDP quanto TCP. A porta padrão é a 514, tanto para UDP quanto TCP.

O conceito de retransmissores (relays) e coletores (collectors) é usado para transmitir e estocar as mensagens: quem gera o evento é o originator ou sender[7], que envia para um relay que por sua vez encaminha a mensagem para o coletor, que estoca a mensagem. Um serviço de syslog contém duas partes, relay e collector. No caso de syslog logando em arquivos locais, não é usado o relay. Originalmente, estes conceitos foram usados no serviço de e-mail, posteriormente aplicados também para syslog.

As mensagens também podem ser enviadas para mais de um retransmissor ou coletor ao mesmo tempo.



Syslog: coletor remoto (loghost) ou local.

Para configurar um loghost syslog, basta editar o arquivo de configuração do syslog, **/etc/rsyslog.conf**. Esta edição é feita de forma diferente em duas máquinas, pois uma assume a função de servidor de logs (loghost) e a outra a função de máquina cliente, que vai enviar as mensagens de log para o loghost.

Deste modo, aqui são usadas duas máquinas diferentes: o servidor loghost no endereço IP 192.168.1.100, representado pelo shell "*loghost#*", e o cliente deste serviço, representado pelo shell "*client-root#*".

PARTE 1: configuração do loghost **rsyslog**.

Configurar o syslog do loghost para coletar as mensagens enviadas pelos relays:

```
loghost# cp  /etc/rsyslog.conf  /etc/rsyslog.conf.SAVE
loghost# vi  /etc/rsyslog.conf
```

```
# provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
```

Na configuração acima, apenas foram retirados os comentários das linhas "$ModLoad imudp" e "$UDPServerRun 514". Isso configura o serviço rsyslog como **coletor**, que atende na porta UDP 514.

NOTA 1: o fato do serviço rsyslog estar configurado para **coletor** não impede que também registre as mensagens geradas localmente.

NOTA 2: o **coletor** foi configurado para usar protocolo de transporte UDP, mas poderia ter

7 Sender: é o processo que gera o evento e cria a mensagem a ser logada.

sido TCP.

Depois, precisa ser reiniciado o serviço syslog. Num sistema SystemV, precisa executar o comando:

```
loghost# /etc/init.d/rsyslog  restart
```

NOTA 1: se fosse um SystemD, o comando seria "**systemctl restart rsyslog**".

NOTA 2: foi dado um **restart** no serviço, pois a configuração envolveu abrir uma porta de serviço (no caso, UDP 514). Normalmente, neste caso um simples reload não é suficiente.

Como pode ser notado pela saída do comando **netstat**, agora a porta UDP 514 está aberta para o relay de mensagens de syslog:

```
loghost# netstat  -na  |  grep  ":514"
udp        0      0 0.0.0.0:514             0.0.0.0:*
```

PARTE 2: configuração do cliente.

Antes de configurar o syslog do cliente, testar com o **logger** para verificar se de fato o loghost (**coletor**) acata mensagens de log via relay do cliente:

```
client-root# logger  -n  192.168.1.100  "teste de aluno - relay de mensagem para o logger"
```

No comando acima, a opção "**-n**" é para informar o nome ou IP do loghost, que neste caso é 192.168.1.100. Também poderia ter sido usado "--server" ao invés de "-n".

No loghost, um tail no **/var/log/syslog** (Ubuntu) deve mostrar o registro remoto desta mensagem:

```
loghost# tail  /var/log/syslog
…
Jul 15 19:06:23 ubuntu root teste de aluno - relay de mensagem para o logger
```

NOTA: num Red Hat, este registro iria para o arquivo /var/log/messages.

Uma vez confirmado que o loghost acata mensagens, configurar o syslog do cliente para fazer relay de mensagens no loghost. Num Ubuntu, editar o arquivo /etc/rsyslog.d/50-default.conf:

```
client-root# cp  /etc/rsyslog.d/50-default.conf  /etc/rsyslog.d/50-default.conf.SAVE
client-root# vi  /etc/rsyslog.d/50-default.conf
```

```
#user.*                 -/var/log/user.log
user.*                  @192.168.1.100
```

NOTA 1: na configuração acima, apenas foi incluída a linha da facilidade "user", que envia as mensagens para o endereço IP do loghost.

NOTA 2: se o loghost estivesse configurado para protocolo de transporte TCP ao invés de UDP, precisaria usar dois arrobas (@@192.168.1.100) ao invés de apenas um.

NOTA 3: se fosse um Red Hat ao invés de Ubuntu, a configuração acima seria feita diretamente no arquivo /etc/rsyslog.conf.

Depois, precisa ser reiniciado o serviço syslog no cliente:

```
client-root# /etc/init.d/rsyslog  restart
```

NOTA: num sistema SystemD, o comando seria "**systemctl restart rsyslog**".

Agora, testar de novo com o **logger**, porém usar a facilidade **user** e enviar uma mensagem para o syslog local:

```
client-root# logger  -p  user.info  "novo teste: envio de mensagem para o syslog local"
```

NOTA: no comando acima, o logger enviou a mensagem para a facilidade **user**, pois é a única que foi configurada para fazer relay no loghost. A criticidade **info** apenas informa do evento.

No loghost, o root dá um tail no arquivo de log **/var/log/userlog**:

```
loghost# tail  /var/log/userlog
…
Jul 15 19:27:10 ubuntu root  novo teste: envio de mensagem para o syslog local
```

A saída do comando acima mostra que a mensagem foi registrada no loghost, ao invés da máquina local.

4.4 - Serviço syslog-ng

Como foi mostrado acima, o serviço de log default (rsyslog) atende bem quando **sender** e **coletor** estão na mesma máquina. Porém, quando o rsyslog é configurado para loghost numa rede com muitos clientes, ele carece de funcionalidade para especificar de qual máquina cliente partiu a mensagem de log.

Então, se a proposta for configurar um loghost, melhor abandonar o rsyslog e partir para outra solução.

O serviço syslog-ng oferece muitas funcionalidades que o rsyslog não tem, como por exemplo classificar e correlacionar mensagens em tempo real.

As vantagens do syslog-ng sobre o tradicional rsyslog, são:

- **filtragem**: filtra as mensagens baseado no conteúdo, além do tradicional par de prioridade "facilidade.nível". Isso permite que que as mensagens geradas em cada equipamento tenham seu próprio arquivo de log;
- **formato de nomes de hosts longos**: a função relay oferecida pelo syslog-ng permite que as mensagens atravessem múltiplos serviços syslog-ng. O formato de nome de host longo, que anota cada serviço syslog-ng, torna fácil encontrar o host de origem e a cadeia de encaminhamento de hosts, mesmo que a mensagem atravesse muitos computadores.

Para configurar um loghost **syslog-ng**, também serão usadas duas máquinas diferentes: o servidor loghost no endereço IP 192.168.1.100, representado pelo shell "*loghost#*", e o cliente deste serviço, representado pelo shell "*client-root#*".

PARTE 1: configuração do loghost **syslog-ng**.

Para instalar o syslog-ng num Ubuntu (ou Debian), comandar:

```
loghost# apt-get  install  syslog-ng-core
```

NOTA 1: num Ubuntu, por default o serviço já é iniciado após a instalação do pacote. É importante notar que antes de levantar o seriço syslog-ng, precisa baixar o rsyslog.

NOTA 2: para instalar o syslog-ng num CentoOS (ou Red Hat), comandar "***yum install syslog-ng syslog-ng-libdbi***".

Depois de instalado, configurar o syslog-ng do loghost para coletar as mensagens enviadas pelos relays:

```
loghost# cp  /etc/syslog-ng/syslog-ng.conf  /etc/syslog-ng/syslog-ng.conf.SAVE
loghost# vi  /etc/syslog-ng/syslog-ng.conf
```

```
@version:3.3                                    # Ubuntu 12
options { owner(root); group(root); perm(0600); dir_perm(0700); create_dirs(yes); };
source s_extern {    udp(port(514)); };         # usar o procolo de transporte UDP e porta 514
destination d_extern { file("/var/log/hosts/$HOST/$YEAR/$MONTH/$DAY/$FACILITY-
$PRIORITY.log"); };
#source s_intern { system(); internal(); };    # para SystemD
source s_intern { file("/proc/kmsg" program-override("kernel: ")); unix-stream ("/dev/log"); internal(); };
destination d_intern { file("/var/log/localhost/$YEAR/$MONTH/$DAY/$FACILITY-$PRIORITY.log"); };
filter f_padrao    { level(notice..emerg); };  # registra tudo exceto debug e info
log { source(s_extern); destination(d_extern); };
log { source(s_intern); filter(f_padrao); destination(d_intern); };
```

Nas configurações acima, temos na primeira linha a versão do syslog-ng.

Nas opções (options), temos:

owner(root); group(root); perm(0600): dono, grupo e permissões dos arquivos de log;
create_dirs (yes): cria automaticamente a estrutura de diretórios;

As linhas ***source s_extern*** e ***destination d_extern*** são usadas para coletar as mensagens enviadas pelos clientes remotos (relay) e armazenar em arquivos de log.

As linhas ***source s_intern*** e ***destination d_intern*** são usadas para coletar as mensagens geradas localmente no loghost e armazenar em arquivos de log.

NOTA: num CentOS, é necessário criar manualmente os diretórios básicos, ***/var/log/hosts*** e ***/var/log/localhost***.

A linha ***filter f_padrao*** está configurada para filtrar as mensagens pelo nível de criticidade e deixar passar apenas níveis de ***notice*** para cima. As criticidades são: ***emerg***, ***alert***, ***crit***, ***err***, ***warning***, ***notice***, ***info*** e ***debug***.

As duas últimas linhas, ***log***, são para armazenar as mensagens em arquivos, relacionando origem (***source***) com destino (***destination***) e aplicando filtragem (***filter***) se for o caso.

Antes de iniciar o serviço syslog-ng, é importante verificar se o serviço rsyslog está rodando, e se for o caso, parar:

```
loghost# ps -ef | grep rsyslog
root     1366    1  0 12:44 ?        00:00:00 /usr/sbin/rsyslogd -n
```

Caso esteja rodando, precisa para parar o rsyslog. Num sistema SystemV, comandar:

```
loghost# /etc/init.d/rsyslog stop
Desligando o agente de log do sistema:                     [ OK ]
```

NOTA 1: num sistema SystemD, o comando seria "**systemctl stop rsyslog**".

NOTA 2: no Ubuntu, na instalação do syslog-ng, o próprio apt-get pára o serviço rsyslog e inicia o syslog-ng.

Depois de parado o rsyslog (se este for o caso), iniciar o syslog-ng:

```
loghost# /etc/init.d/syslog-ng start
Iniciando o syslog-ng:                                     [ OK ]
```

NOTA: num sistema SystemD, o comando seria "**systemctl start syslog-ng**".

Com o seviço syslog-ng instalado e configurado, verificar se a porta 514 UDP agora está aberta no loghost:

```
loghost# netstat -na | grep ":514"
udp        0      0 0.0.0.0:514             0.0.0.0:*               OUÇA
```

Observar que, pela configuração do syslog-ng, agora os registros de log locais vão para o diretório **/var/log/localhost**, e incluem subdiretórios para o ano, mês e dia. Por exemplo, para o ano

de 2017, mês de julho, dia 16, os arquivos de logs ficam em **/var/log/localhost/2017/07/16/**.

No loghost, testar com o logger para verificar se está registrando os eventos gerados localmente:

```
loghost# logger  "teste local no loghost"
loghost# tail  -1  /var/log/localhost/2017/07/16/user-notice.log
Jul  16  13:39:33  s_intern@loghost  aluno: teste local no loghost
```

Depois disso, instalar e configurar o syslog-ng no cliente, para logar remotamente no loghost.

PARTE 2: configuração do **cliente syslog-ng**.

Antes de configurar o syslog-ng do cliente, testar com o **logger** para verificar se de fato o loghost (**coletor**) acata mensagens de log via relay do cliente:

```
client-root# logger  -n  192.168.1.100  "teste de aluno - relay de mensagem para o syslog-ng"
```

No comando acima, a opção "**-n**" é para informar o nome ou IP do loghost, que neste caso é 192.168.1.100. Também poderia ter sido usado "--server" ao invés de "-n".

No loghost syslog-ng, deverá ser registrado o log da mensagem:

```
loghost# tail  -1  /var/log/hosts/client-root/2017/07/16/user-notice.log
Jul  16  18:01:10  client-root 1 2017-07-16T18:01:10.125587-03:00 client-root root - - [timeQuality tzKnown="1" isSynced="1" syncAccuracy="546500"] teste de aluno - relay de mensagem para o syslog-ng
```

A instalação no cliente é idêntica ao caso do loghost. Para instalar o syslog-ng num Ubuntu (ou Debian), comandar:

```
client-root# apt-get  install  syslog-ng-core
```

NOTA 1: num Ubuntu, por default o serviço já é iniciado após a instalação do pacote. É importante notar que antes de levantar o serviço syslog-ng, precisa baixar o rsyslog.

NOTA 2: para instalar o syslog-ng num CentoOS (ou Red Hat), comandar "***yum install syslog-ng syslog-ng-libdbi***".

Configurar o syslog-ng do cliente para fazer relay das mensagens no loghost. No exemplo abaixo, foi usado um Ubuntu 12 (SystemV):

```
client-root# cp  /etc/syslog-ng/syslog-ng.conf  /etc/syslog-ng/syslog-ng.conf.SAVE
client-root# vi  /etc/syslog-ng/syslog-ng.conf
```

```
@version:3.3                                                    # Ubuntu 12
destination  loghost  { udp("192.168.0.23"  port(514)); };      # O endereço IP do loghost.
source  src  { file("/proc/kmsg" program-override("kernel: ") flags(kernel) keep_timestamp(no)); unix-
stream ("/dev/log"); internal(); };
#source  src { system(); internal(); };                         # para SystemD
filter  f_padrao    { level(notice..emerg); };                  # registra tudo exceto debug e info
log  { source(src); filter(f_padrao); destination(loghost); };  # envia tudo para o loghost.
```

NOTA 1: a configuração do cliente é semelhante às configurações de registros de log locais do loghost, com a diferença que não envia para arquivos e sim para o coletor.

NOTA 2: o cliente pode enviar as mensagens tanto com protocolo de transporte TCP quanto UDP. Porém, deve ser o mesmo protocolo configurado no loghost.

Caso esteja rodando, precisa para parar o rsyslog no cliente:

```
client-root# /etc/init.d/rsyslog  stop
Desligando o agente de log do sistema:                          [  OK  ]
```

Depois, se este for o caso, iniciar o syslog-ng no cliente:

```
client-root# /etc/init.d/syslog-ng  start
Iniciando o syslog-ng:                                          [  OK  ]
```

Para testar, usar o logger para enviar uma mensagem do cliente para o loghost:

```
client-root# logger  "teste de envio do cliente para o loghost syslog-ng"
```

No loghost, verificar que a mensagem foi registrada no log:

```
loghost# tail  -1  /var/log/hosts/client-root/2017/07/16/user-notice.log
Jul  16  18:03:20  client-root  root:  teste de envio do cliente para o loghost syslog-ng
```