Deprivilege Root access of a daemon

By:
Ram Jangir (CS21M517)
Sneha M R (CS21M522)

What is sysvinit?

- It is the first process (init process pid=0)started by the kernel when you boot up any Linux or Unix system. It means other processes are its child in one or the other way.
- Sysvinit process continues to run and waits for special commands like ‘shutdown’, which are used to shut down a Linux system.

Disadvantages:

- Sysvinit remained a perfect system to bring up and shutdown Linux-based systems. But as time passed by, the system became slow and inflexible, especially for modern-day computers/systems.

In 2010 systemd was proposed to replace the widely used sysvinit system.

Advantages of systemd over sysvinit:

Services are started in parallel instead of serially, which reduces boot-up time.

Service configuration is easier theough service files.

It has more robust control and debugging like journalctl.

It is compatible with **Sysvinit.**

Service File Options

How to Improve:

- **Service file Options:**
  - User= and Group= options in systemd service files
  - Allows launching of daemons as non-root users/groups without needing setuid/setgid capabilities.

- Systemd-tmpfiles
  - Daemon that runs on startup
  - Reads configuration files supplied by /etc.tmpfiles.d/*.conf and creates/modifies files after bootup.
  - Runs as root so your daemon doesn’t have to.

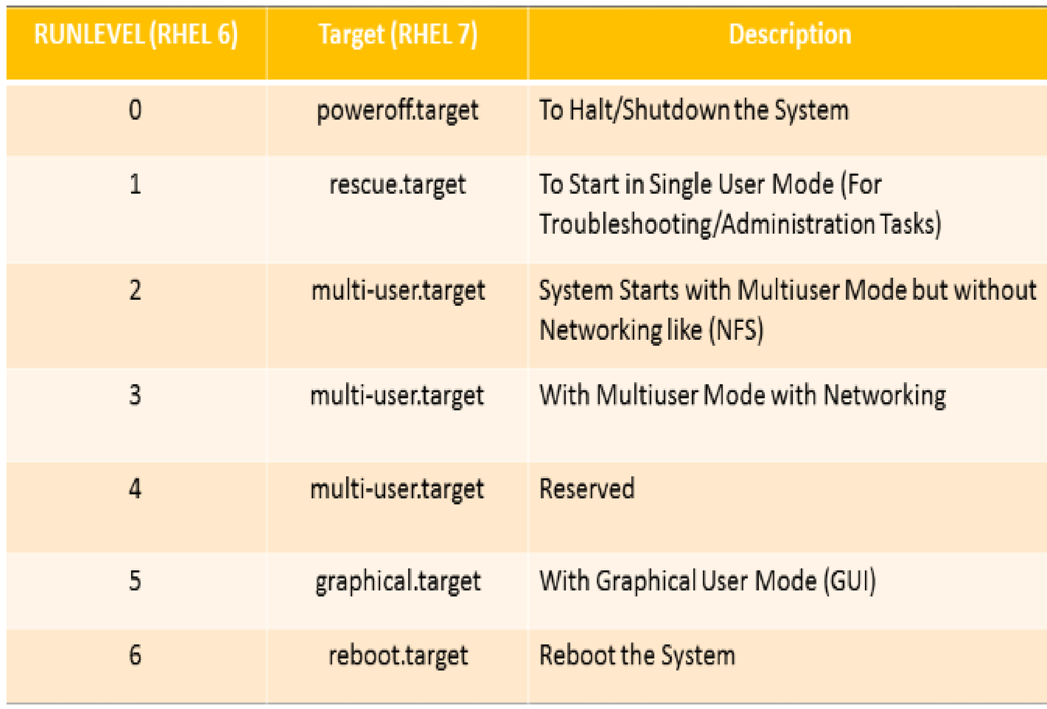
- Udev
  - Continuously running daemon
  - Updates sysfs permissions once at bootup and monitors uevents for any dynamically created nodes from the kernel. If new nodes are added – this daemon will update the permissions

Example service file

```
cat /etc/systemd/system/app.service

[Unit]
SourcePath=/local/mnt/workspace/SERVICE_FILE/app
Description= APP Service file


[Service]
User=radio
Group=radio


ExecStart=/local/mnt/workspace/SERVICE_FILE/app
ExecStop= /usr/bin/kill -9 app

SupplementaryGroups=diag inet net_admin system wifi netdev net_raw kmsg
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_RAW CAP_NET_BIND_SERVICE CAP_SETUID CAP_SETGID
AmbientCapabilities=CAP_NET_ADMIN CAP_NET_RAW CAP_NET_BIND_SERVICE CAP_SETUID CAP_SETGID
```

Detailed service file:

```
[Unit]
SourcePath=/local/mnt/workspace/SERVICE_FILE/app
Description= APP Service file

[Service]
User=radio
Group=radio

ExecStart=/local/mnt/workspace/SERVICE_FILE/app
ExecStop= /usr/bin/kill -9 app
RemainAfterExit=
Type=
Wants=
After=
Before=
Part of=
WantedBy=
RequiredBy=
Restart-on-failure=yes

SupplementaryGroups=diag inet net_admin system wifi netdev net_raw kmsg
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_RAW CAP_NET_BIND_SERVICE CAP_SETUID CAP_SETGID
AmbientCapabilities=CAP_NET_ADMIN CAP_NET_RAW CAP_NET_BIND_SERVICE CAP_SETUID CAP_SETGID

[Install]
Wantedby=multi-user.target
```



RUNLEVEL (RHEL 6)	Target (RHEL 7)	Description
0	poweroff.target	To Halt/Shutdown the System
1	rescue.target	To Start in Single User Mode (For Troubleshooting/Administration Tasks)
2	multi-user.target	System Starts with Multiuser Mode but without Networking like (NFS)
3	multi-user.target	With Multiuser Mode with Networking
4	multi-user.target	Reserved
5	graphical.target	With Graphical User Mode (GUI)
6	reboot.target	Reboot the System

Systemd Commands:

- systemctl start app # start the service. (app is the service file name)
- systemctl stop app # Stop the service.
- systemctl restart app # Restart the service.
- systemctl status app # Get the status of the service
- journalctl # Get all the logs
- journalctl –u app # Get the logs of the particular service
- journalctl –u app –n 10 # Get the logs of the particular service, **10 lines**
- systemctl enable app #Run everytime after bootup
- systemctl disable app #Dont run after bootup.

Challenge 1)

- ***Procfs entries and network related commands are impacted:***

- */proc/sys/net/ipv6/conf/disable_ipv6*
- */proc/sys/net/ipv4/ip_forward*
- */proc/sys/net/netfilter/nf_conntrack_udp_timeout_stream*
- */proc/sys/debug/sfe/packet_stats_on*

- ***Brctl addif wlan0 bridge0***
- ***iw dev wlan0 set 4addr on***
- ***Ifconfig wlan 0 up , down***
- ***Vconfig <iface name>***

- ***Solution:***
- ***Giving the NET_ADMIN and NET_RAW capabilities***

Systemd-tmpfiles

How to Improve:

- Service file Options:
  - User= and Group= options in systemd service files
  - Allows launching of daemons as non-root users/groups without needing setuid/setgid capabilities.
- **Systemd-tmpfiles:**
  - Daemon that runs on startup
  - Reads configuration files supplied by /etc.tmpfiles.d/*.conf and creates/modifies files after bootup.
  - Runs as root so your daemon doesn’t have to.
- Udev
  - Continuously running daemon
  - Updates sysfs permissions once at bootup and monitors uevents for any dynamically created nodes from the kernel. If new nodes are added – this daemon will update the permissions

Systemd-tmpfiles:

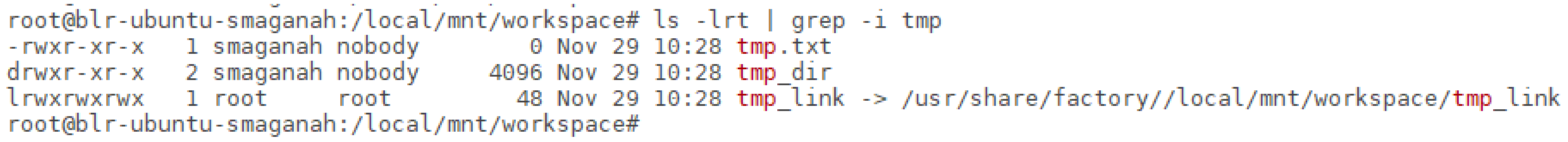
$cat /etc/tmpfiles.d/test.conf

```
f   /local/mnt/workspace/tmp.txt 0755 smaganah nobody - -
d  /local/mnt/workspace/tmp_dir 0755 smaganah nobody - -
L  /local/mnt/workspace/tmp_link 0755 smaganah nobody - -
```

Command:

```
$systemd-tmpfiles --create
```

Output:



```
root@blr-ubuntu-smaganah:/local/mnt/workspace# ls -lrt | grep -i tmp
-rwxr-xr-x   1 smaganah nobody        0 Nov 29 10:28 tmp.txt
drwxr-xr-x   2 smaganah nobody     4096 Nov 29 10:28 tmp_dir
lrwxrwxrwx   1 root     root         48 Nov 29 10:28 tmp_link -> /usr/share/factory//local/mnt/workspace/tmp_link
root@blr-ubuntu-smaganah:/local/mnt/workspace#
```

Challenge 2)

- ***/var/run and /tmp are impacted:***

- /tmp/statefdbtable.txt
- /tmp/session_token
- /tmp/brctl.tmp
- /var/run/dnsmasq_socket

- ***Solution:***
- ***Create /var/run/data and /tmp/data , the data folders have radio:radio permission. Hence only Private daemons can access it. Use systemd-tmpfiles to create these folders.***

Challenge 3)

/etc/data is impacted:

- ***Solution:***
- ***The config files used by the private daemons is present in /etc/data should be changed to radio:radio permission.***

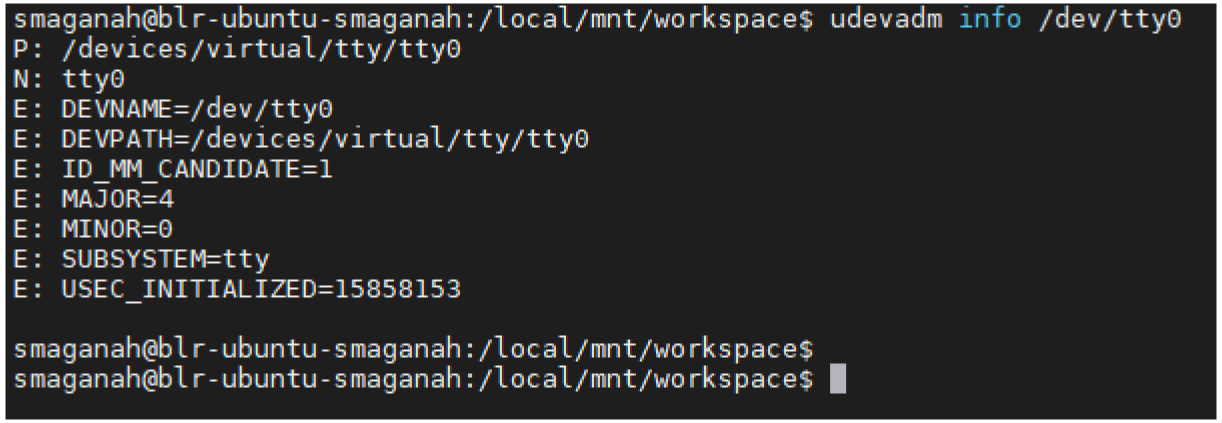
- ***The config file which are used by both private daemon and opensource daemon (eg: dnsmasq.conf) should be changed to root:radio permission so that both Private-daemon (runs as radio) and open-source daemon (runs as root) should be able to edit it.***

Udev events

How to Improve:

- Service file Options:
  - User= and Group= options in systemd service files
  - Allows launching of daemons as non-root users/groups without needing setuid/setgid capabilities.
- systemd-tmpfiles
  - Daemon that runs on startup
  - Reads configuration files supplied by /etc.tmpfiles.d/*.conf and creates/modifies files after bootup.
  - Runs as root so your daemon doesn’t have to.
- **Udev:**
  - Continuously running daemon
  - Updates sysfs permissions once at bootup and monitors uevents for any dynamically created nodes from the kernel. If new nodes are added – this daemon will update the permissions

- **udevadm info /dev/tty0**


```
smaganah@blr-ubuntu-smaganah:/local/mnt/workspace$ udevadm info /dev/tty0
P: /devices/virtual/tty/tty0
N: tty0
E: DEVNAME=/dev/tty0
E: DEVPATH=/devices/virtual/tty/tty0
E: ID_MM_CANDIDATE=1
E: MAJOR=4
E: MINOR=0
E: SUBSYSTEM=tty
E: USEC_INITIALIZED=15858153

smaganah@blr-ubuntu-smaganah:/local/mnt/workspace$
smaganah@blr-ubuntu-smaganah:/local/mnt/workspace$
```

Commands to trigger Udev event:
udevadm info /dev/tty0
udevadm control --reload-rules
udevadm trigger

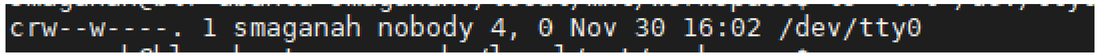
$cat /etc/udev/rules.d/udev_rules.rules

```
SUBSYSTEM=="tty" RUN+="/usr/bin/udev_script.sh"
```

$cat /usr/bin/udev_script.sh

```
#!/bin/sh
chown -h smaganah.nobody /dev/tty0
```

Output:


```
crw--w----. 1 smaganah nobody 4, 0 Nov 30 16:02 /dev/tty0
```

Challenge 4)

Printing logs to /dev/kmsg is impacted:

Solution:
/var/log/messages should be used instead of /dev/kmsg

/dev/ttyX is impacted:

Solution:
/dev/ttyX permission should be changed using udev rules.

System DBUS (SDBUS)

System Dbus:

- Sdbus allows depivileged process to start a process in root:root (user:group)

Example:

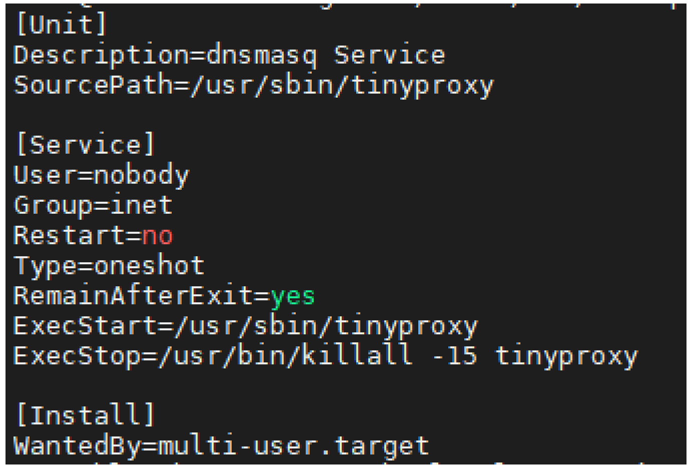
DBUS API to start the daemon:

```
 r = sd_bus_call_method(bus,
                        "org.freedesktop.systemd1",
                         "/org/freedesktop/systemd1",
                         "org.freedesktop.systemd1.Manager",
                         "StartUnit", //specify to start the process. Similary StopUnit, RestartUnit
                          &error,
                          &m,
                          "ss",
                          “tinyproxy.service”, //specify which service to start or stop
                          "replace");
```

It will start the tinyproxy service.

Sample Open source daemon service file:

$cat /etc/init.d/tinyproxy.service



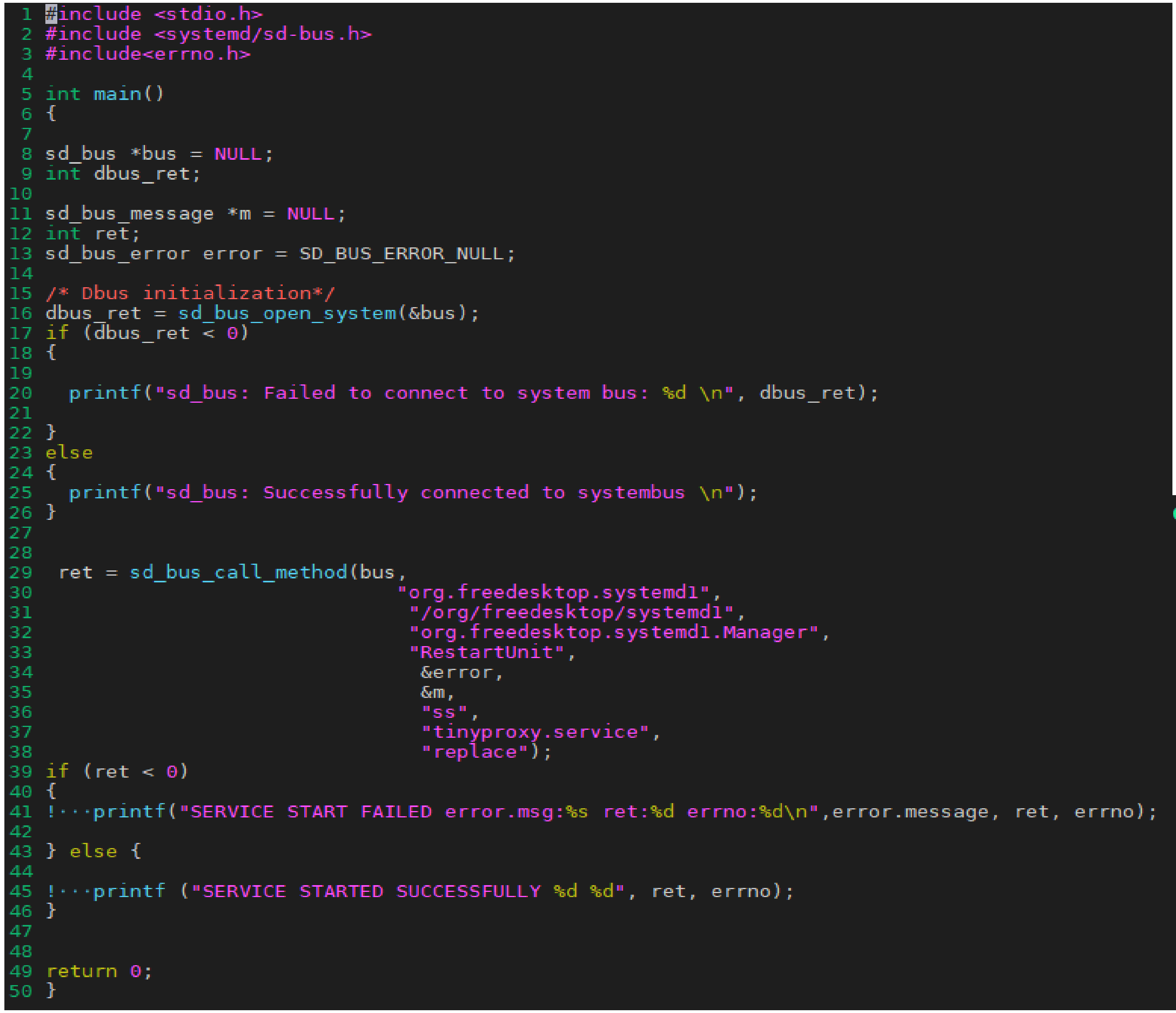
```
[Unit]
Description=dnsmasq Service
SourcePath=/usr/sbin/tinyproxy

[Service]
User=nobody
Group=inet
Restart=no
Type=oneshot
RemainAfterExit=yes
ExecStart=/usr/sbin/tinyproxy
ExecStop=/usr/bin/killall -15 tinyproxy

[Install]
WantedBy=multi-user.target
```

Installation:

sudo apt-get install tinyproxy

sudo apt-get install libsystemd-dev

Sample Program:

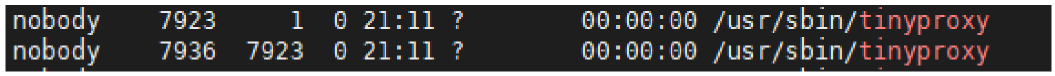


```c
#include <stdio.h>
#include <systemd/sd-bus.h>
#include<errno.h>

int main()
{

sd_bus *bus = NULL;
int dbus_ret;

sd_bus_message *m = NULL;
int ret;
sd_bus_error error = SD_BUS_ERROR_NULL;

/* Dbus initialization*/
dbus_ret = sd_bus_open_system(&bus);
if (dbus_ret < 0)
{

  printf("sd_bus: Failed to connect to system bus: %d \n", dbus_ret);

}
else
{
  printf("sd_bus: Successfully connected to systembus \n");
}


 ret = sd_bus_call_method(bus,
                           "org.freedesktop.systemd1",
                            "/org/freedesktop/systemd1",
                            "org.freedesktop.systemd1.Manager",
                            "RestartUnit",
                             &error,
                             &m,
                             "ss",
                             "tinyproxy.service",
                             "replace");
if (ret < 0)
{
!···printf("SERVICE START FAILED error.msg:%s ret:%d errno:%d\n",error.message, ret, errno);

} else {

!···printf ("SERVICE STARTED SUCCESSFULLY %d %d", ret, errno);
}


return 0;
}
```

Compile:
$gcc sd_bus.c –lsystemd

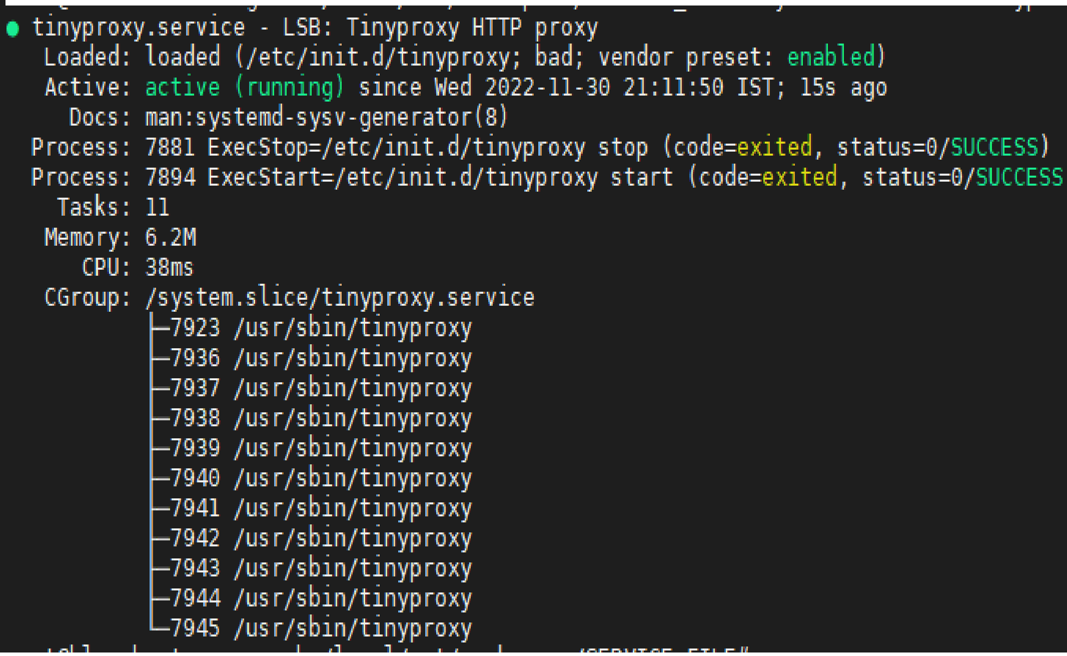
Output:
$./a.out
sd_bus: Successfully connected to systembus
SERVICE STARTED SUCCESSFULLY 1 11

$ps -ef | grep -i tinyproxy



```
nobody     7923      1  0 21:11 ?        00:00:00 /usr/sbin/tinyproxy
nobody     7936   7923  0 21:11 ?        00:00:00 /usr/sbin/tinyproxy
```

$systemctl status tinyproxy

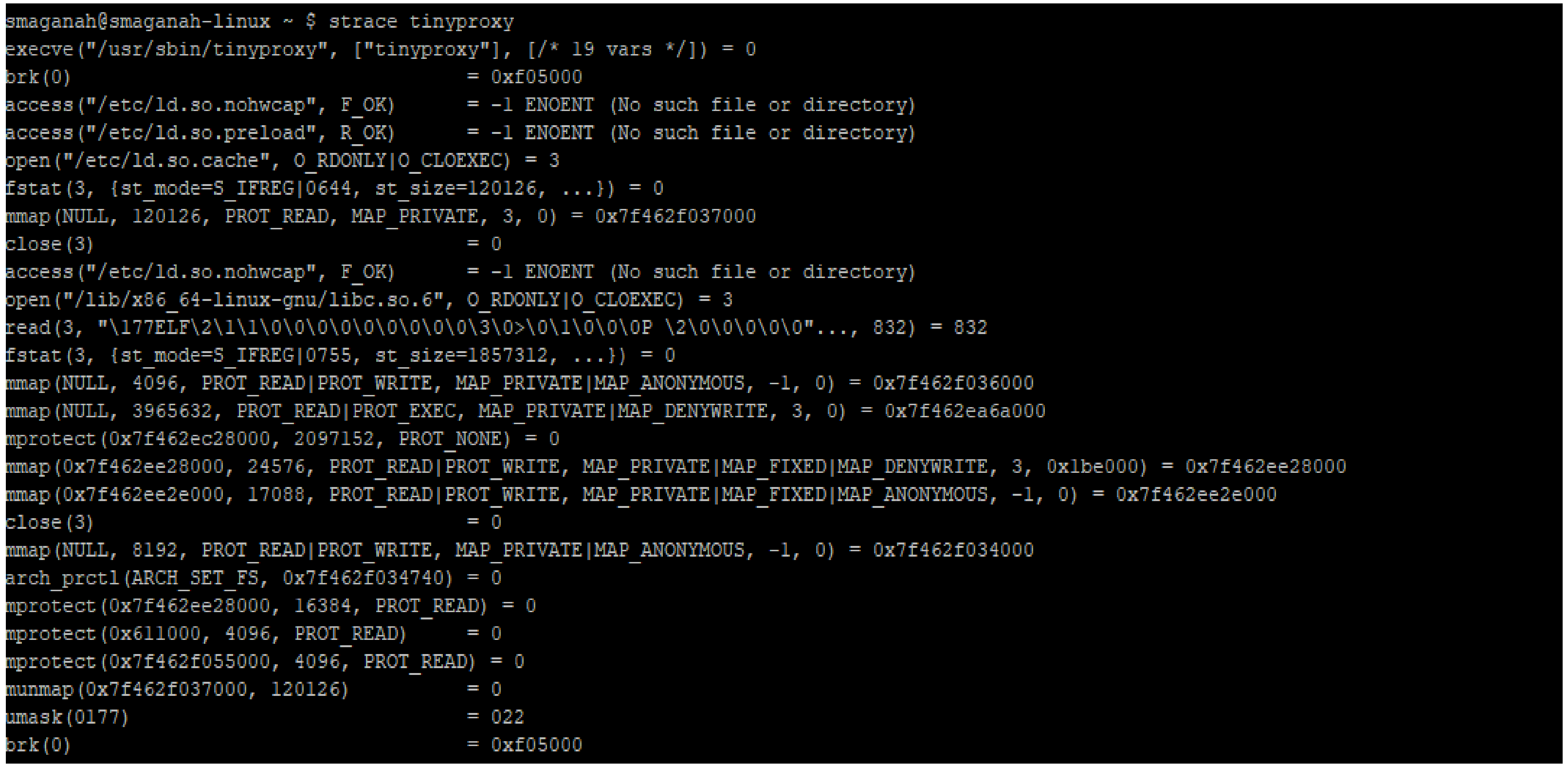


```
● tinyproxy.service - LSB: Tinyproxy HTTP proxy
   Loaded: loaded (/etc/init.d/tinyproxy; bad; vendor preset: enabled)
   Active: active (running) since Wed 2022-11-30 21:11:50 IST; 15s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 7881 ExecStop=/etc/init.d/tinyproxy stop (code=exited, status=0/SUCCESS)
  Process: 7894 ExecStart=/etc/init.d/tinyproxy start (code=exited, status=0/SUCCESS
    Tasks: 11
   Memory: 6.2M
      CPU: 38ms
   CGroup: /system.slice/tinyproxy.service
           ├─7923 /usr/sbin/tinyproxy
           ├─7936 /usr/sbin/tinyproxy
           ├─7937 /usr/sbin/tinyproxy
           ├─7938 /usr/sbin/tinyproxy
           ├─7939 /usr/sbin/tinyproxy
           ├─7940 /usr/sbin/tinyproxy
           ├─7941 /usr/sbin/tinyproxy
           ├─7942 /usr/sbin/tinyproxy
           ├─7943 /usr/sbin/tinyproxy
           ├─7944 /usr/sbin/tinyproxy
           └─7945 /usr/sbin/tinyproxy
```

Debugging with strace:

Usage:
Strace <binary name>

Eg: strace tinyproxy



```
smaganah@smaganah-linux ~ $ strace tinyproxy
execve("/usr/sbin/tinyproxy", ["tinyproxy"], [/* 19 vars */]) = 0
brk(0)                                  = 0xf05000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=120126, ...}) = 0
mmap(NULL, 120126, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f462f037000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P \2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1857312, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f462f036000
mmap(NULL, 3965632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f462ea6a000
mprotect(0x7f462ec28000, 2097152, PROT_NONE) = 0
mmap(0x7f462ee28000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1be000) = 0x7f462ee28000
mmap(0x7f462ee2e000, 17088, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f462ee2e000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f462f034000
arch_prctl(ARCH_SET_FS, 0x7f462f034740) = 0
mprotect(0x7f462ee28000, 16384, PROT_READ) = 0
mprotect(0x611000, 4096, PROT_READ)     = 0
mprotect(0x7f462f055000, 4096, PROT_READ) = 0
munmap(0x7f462f037000, 120126)          = 0
umask(0177)                             = 022
brk(0)                                  = 0xf05000
```

Future Work:

- Implement SELinux

Thank You