Using kAFS on Linux for Network Home Directories

Jonathan Billings
University of Michigan,
College of Engineering, CAEN

Topics for today

- Background
- AFS in the Linux Kernel (kAFS)
- kAFS features
- kAFS vs systemd
- Why use kAFS?
- Future of kAFS
- Questions

AFS?

- Distributed network filesystem
- Allows replication of data across multiple servers
- Supports user authentication
- Provides access control lists for managing user and group permissions

History of AFS

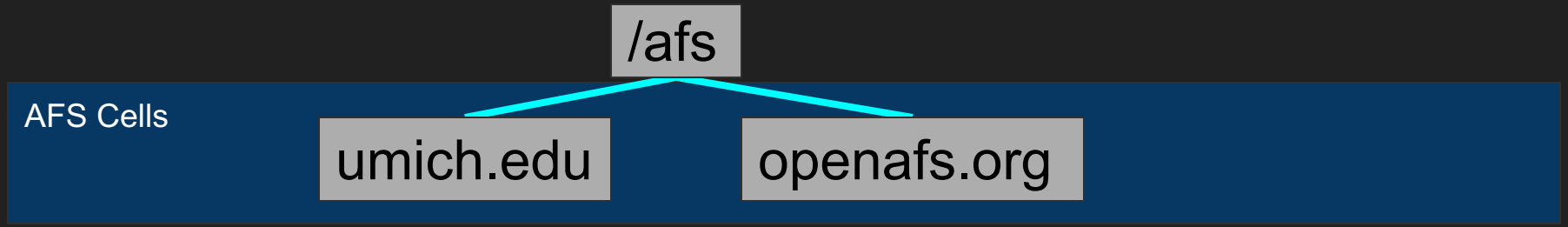
- The Andrew File System
  - Developed at Carnegie Mellon University in collaboration with Transarc as the VICE file system as part of the Andrew Project in 1985
  - Later sold as a commercial product by Transarc (acquired by IBM)
- OpenAFS
  - IBM opened the source to the Transarc product under an IBM Public License in 2000
  - The source is now managed by the OpenAFS foundation

What is AFS?

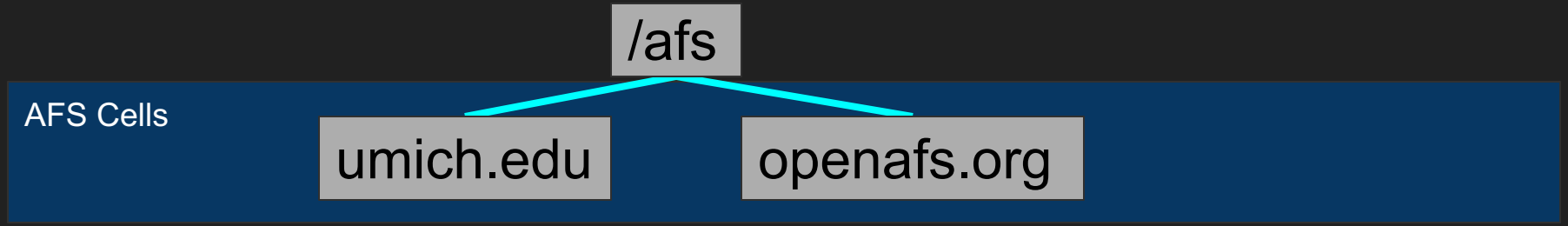
- Runs on top of RXRPC
- Global file system made up of a collection of cells
- Data is stored in volumes served by AFS file servers
- Uses Kerberos for authentication
- Paths appear as /afs/cellname/data.txt

What is RXRPC?

- Runs over UDP
- Remote Procedure Call service
- Security with Kerberos 5
- AFS uses RX for all its services
  - Services correspond to a service number
  - Service numbers defined in Rx_services registry
- It is possible to run other services over RXRPC

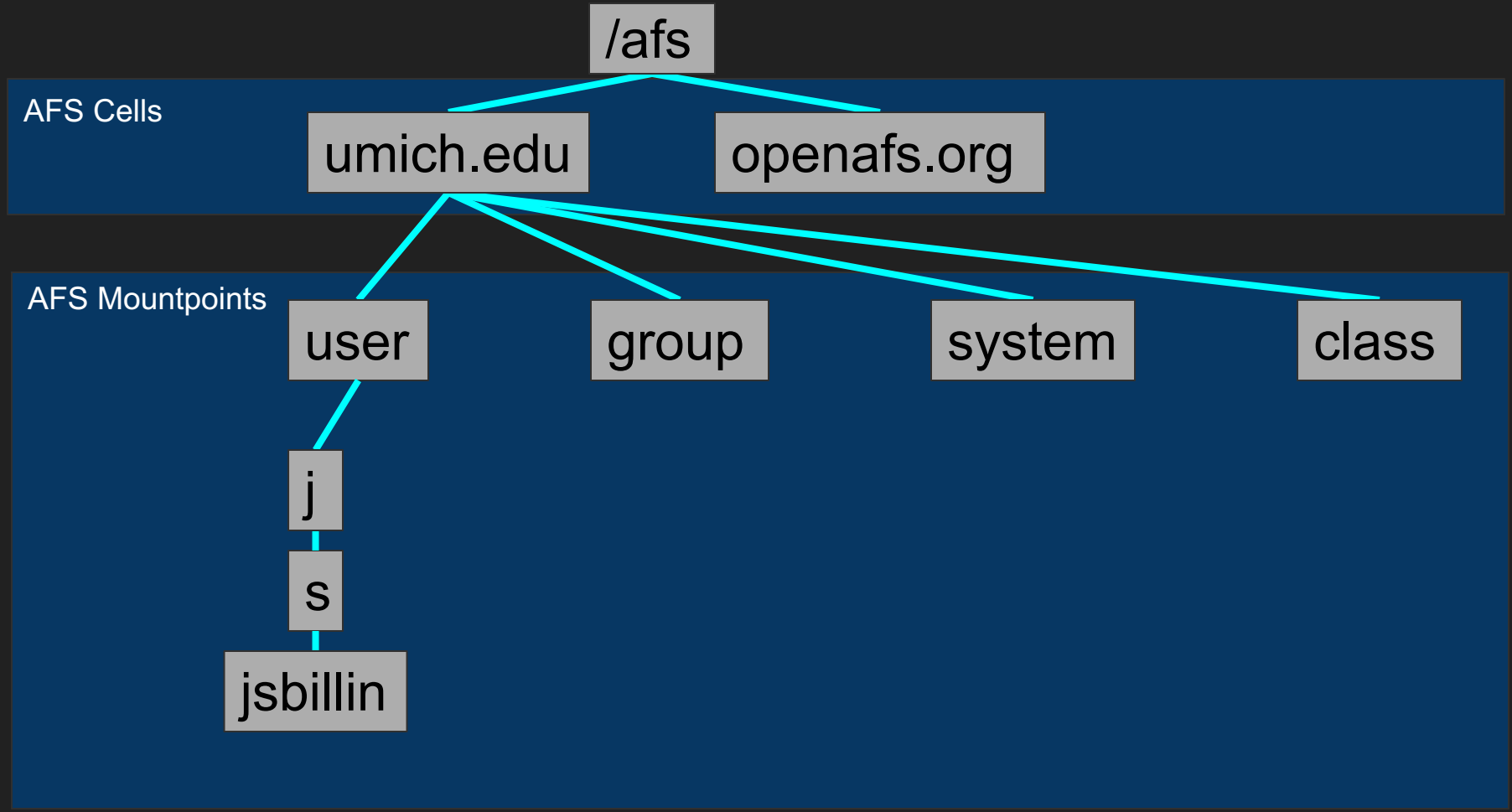


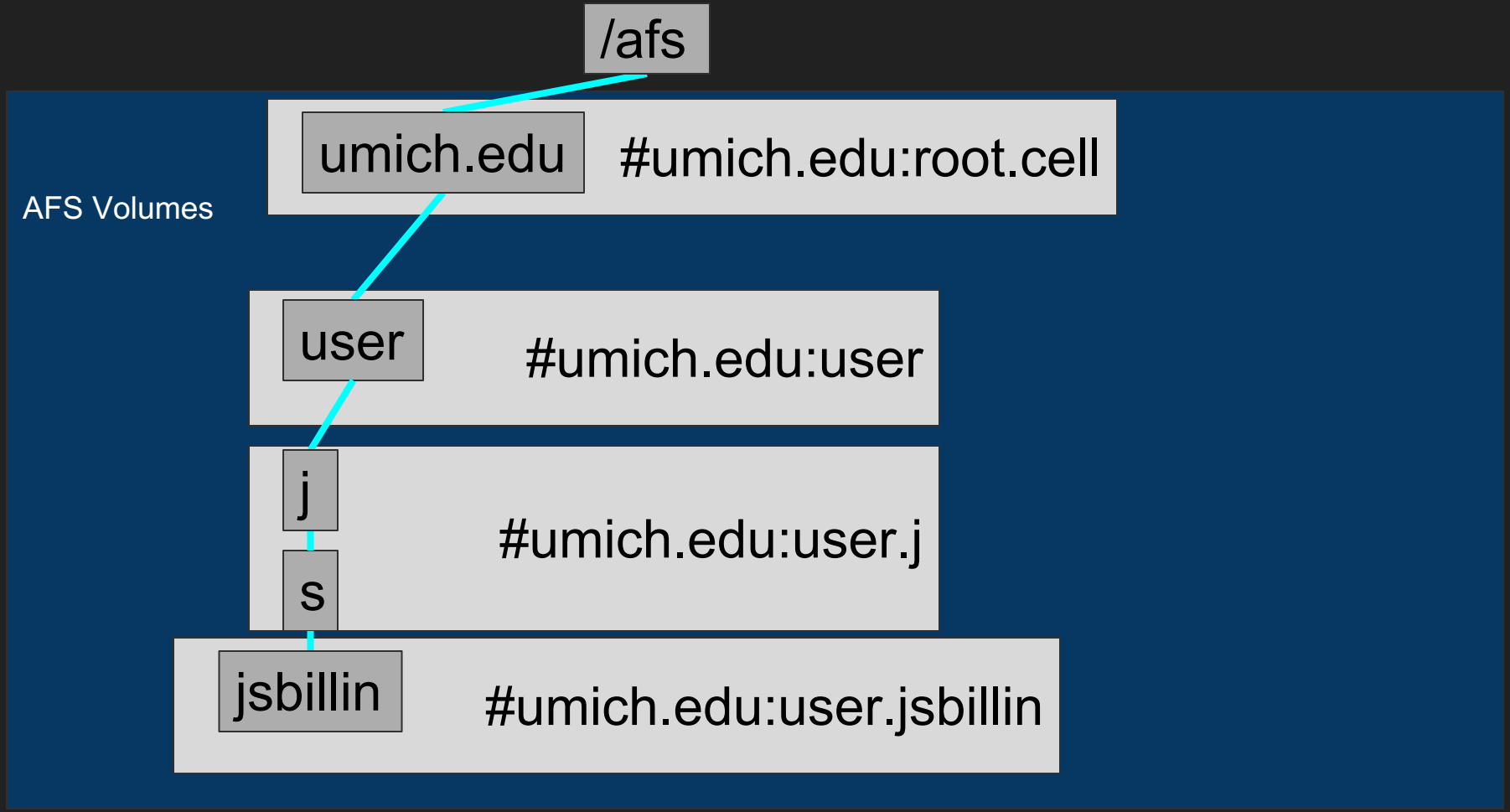

```
CellServDB:
>umich.edu              #University of Michigan - Campus
141.211.1.32               #fear.ifs.umich.edu
141.211.1.33               #surprise.ifs.umich.edu
141.211.1.34               #ruthless.ifs.umich.edu
```




```
DNS:
% dig +short umich.edu -t afsdb
1 fear.ifs.umich.edu.
1 surprise.ifs.umich.edu.
1 ruthless.ifs.umich.edu.
```

(Can also use SRV records)


/afs
AFS Cells
umich.edu
openafs.org
AFS Mountpoints
user
group
system
class
j
s
jsbillin


/afs
umich.edu
#umich.edu:root.cell
AFS Volumes
user
#umich.edu:user
j
s
#umich.edu:user.j
jsbillin
#umich.edu:user.jsbillin

Global File System

```
% ls /afs/umich.edu/
class/  group/  README  system/  um/  user/

% ls /afs/openafs.org
archive/  cvs/    doc/  local/  project/    service/
software/  user/  www/

% wc -l /afs/umich.edu/user/j/s/jsbillin/.plan
13 /afs/umich.edu/user/j/s/jsbillin/.plan
```

Magic @sys expansion

- AFS clients have a concept of a “sysname”
- Can be one or more strings
  - By default on 64-bit x86_64 is “amd64_linux24”
  - Can add other sysnames such as “amd64_fedora_31”
- Create symlink or environment variable that includes @sys in path
- Kernel automatically tries to substitute each member of the sysname array, in order
- Very useful on multiple OSs, stages or classes

Magic @sys expansion in symlinks

For example, I have the hostname in the sysname array

```
$ hostname
rhea
$ fs sys
Current sysname list is 'rhea' 'amd64_fedora_31'

$ echo “I am $HOSTNAME” > testfile.$HOSTNAME
$ ls -ago testfile*
lrwxr-xr-x. 1 13 Jan 18 12:01 testfile -> testfile.@sys
-rw-r--r--. 1 31 Jan 18 12:00 testfile.rhea
-rw-rw-r--. 1 33 Jan 18 12:00 testfile.tethys
$ cat testfile
I am rhea
```

Why use AFS?

- Atomic volume moves and replication
- Strong identity management tied to Kerberos
- Open source and commercial clients for Windows, macOS, Linux, BSD
- Supports collaborating across multiple cells

Why AFS isn’t as widely adopted

- Complex server infrastructure
- Can’t simply export existing dataset
- Client doesn’t support all filesystem objects such as Unix domain sockets
- Open source AFS client used to require an out of tree kernel module

AFS in the Linux Kernel

AFS in the Linux Kernel

- First introduced into the kernel in 2002
- Only had read support and manual mounts
- Improvements in the past two years have made it usable for end users
- Enabled using these kernel build config options
  - CONFIG_AFS_FS=m
  - CONFIG_AFS_FSCACHE=y
  - CONFIG_AFS_DEBUG=y
  - CONFIG_AF_RXRPC=m
  - CONFIG_AF_RXRPC_IPV6=y
  - CONFIG_AF_RXRPC_DEBUG=y
  - CONFIG_RXKAD

Linux features used by kAFS

- In the Kernel
  - FS-Cache
  - Kernel keyrings
  - Kernel tracepoints (for debugging)
  - In-kernel cryptology functions
  - Dynamic automount
- Keyutil
  - For storing AFS tokens in kernel keyring
  - Uses request-key to handle callback requests from the kernel

kAFS on a Linux system

- Set up a mountpoint for /afs

```
# /usr/lib/systemd/system/afs.mount
[Unit]
Description=kAFS Dynamic Root mount
ConditionPathExists=/afs
Wants=kafs-config.service

[Mount]
What=none
Where=/afs
Type=afs
Options=_netdev,dyn

[Install]
WantedBy=remote-fs.target
```

AFS Volumes on kAFS

```
% df -h /afs/umich.edu
Filesystem            Size  Used Avail Use% Mounted on
#umich.edu:root.cell  4.9M   17K  4.9M   1% /afs/umich.edu

% df -h /afs/umich.edu/user
Filesystem    Size  Used Avail Use% Mounted on
#umich.edu:user 1000K   38K  962K   4% /afs/umich.edu/user

% df -h /afs/umich.edu/user/j/s/jsbillin
Filesystem                  Size  Used Avail Use% Mounted on
%umich.edu:user.jsbillin    10G  8.9G  1.2G  89%
/afs/umich.edu/user/j/s/jsbillin
```

Volume Mountpoints

```
% readlink /afs/umich.edu/user/j
#user.j.

% ls -d /afs/umich.edu/user/j/
/afs/umich.edu/user/j/

% grep umich.edu /proc/mounts
#umich.edu:root.cell /afs/umich.edu afs rw,relatime 0 0
#umich.edu:user /afs/umich.edu/user afs rw,relatime 0 0
#umich.edu:user.j /afs/umich.edu/user/j afs rw,relatime 0 0
```

Read vs. Read/Write Volume Mountpoints

```
% df /afs/umich.edu/user
Filesystem    1K-blocks  Used Available Use% Mounted on
#umich.edu:user      1000       38         962   4%
/afs/umich.edu/user

% df /afs/umich.edu/user/j/s/jsbillin
Filesystem           1K-blocks     Used Available Use% Mounted
on
%umich.edu:user.jsbillin  10485760 9065201   1420559  87%
/afs/umich.edu/user/j/s/jsbillin
```

```
% afs vos examine root.cell
root.cell                          536870918 RW     17 K On-line
        141.211.212.61 /vicepa
        RWrite  536870918 ROnly  536870919 Backup  536870920
        MaxQuota         5000 K
        Creation Wed Jun 16 19:57:56 1993
        Copy     Mon Aug 19 19:29:18 2019
        Backup   Mon Jan 06 17:15:01 2020
        Last Access Tue Jan 07 16:26:46 2020
        Last Update Fri Jan 05 11:37:33 2007
        794 accesses in the past day (i.e., vnode references)

        RWrite: 536870918         ROnly: 536870919 Backup: 536870920
        number of sites -> 4
        server afsprs25.afs.storage.umich.edu partition /vicepa RW Site
        server afsprs13.afs.storage.umich.edu partition /vicepa RO Site
        server afsprs14.afs.storage.umich.edu partition /vicepa RO Site
        server afsprs25.afs.storage.umich.edu partition /vicepa RO Site
```

```
% afs vos examine user.jsbillin
user.jsbillin                    1957144488 RW    9240189 K On-line
        141.211.212.58 /vicepc
        RWrite 1957144488 ROnly           0 Backup 1957144490
        MaxQuota   10485760 K
        Creation Fri Jun 13 11:01:26 2008
        Copy     Thu Aug 29 17:15:36 2019
        Backup   Mon Jan 06 18:15:18 2020
        Last Access Tue Jan 07 18:03:30 2020
        Last Update Tue Jan 07 17:52:23 2020
        56391 accesses in the past day (i.e., vnode references)

        RWrite: 1957144488       ROnly: 1957144489Backup: 1957144490
        number of sites -> 1
        server afsprs22.afs.storage.umich.edu partition /vicepc RW Site
```

Authentication in kAFS

```
% kinit
Password for jsbillin@UMICH.EDU:
% aklog-kafs --verbose
CELL umich.edu
REALM UMICH.EDU
PRINC afs/umich.edu@UMICH.EDU
plen=379 tklen=355 rk=24
% keyctl show @s
Keyring
 845055086 --alswrv  263726  1000  keyring: _ses
 895678745 --als-rv  263726  1000   \_ rxrpc: afs@umich.edu
```

kAFS Features not in OpenAFS

- Uses FS-Cache for caching filesystem objects instead of separate caching daemon
- Supports IPv6 RX connections (when available)
- Each volume is its own mount
  - Quota of AFS volume is reflected in df output
- Supports event tracing
  - /sys/kernel/debug/tracing/events/afs
  - /sys/kernel/debug/tracing/events/rxrpc
- Does not use pioctl()

kAFS vs. systemd

- systemd has no problem launching kAFS client
- Issues arise when logging into a system
  - User logs in through PAM stack (ssh, gdm, login, etc.)
  - Systemd starts a “systemd --user” process on behalf of the user
- The systemd --user process
  - Is not a child of the login process
  - Does not inherit any environment variables or session keyring
  - Attempts to read systemd user units out of $HOME/.config/systemd/

Kerberos vs. systemd

- Home directory storage that uses krb5 authentication:
  - NFSv4 with sec=krb5
  - SMB with sec=krb5
  - OpenAFS and AuristorFS
  - Linux kAFS

Kerberos vs. systemd

- The recommended configuration of Kerberos 5 is to have session-based Kerberos tickets
- Ticket caches are created as part of login and defined by an environment variable, KRB5CCNAME

```
$ echo $KRB5CCNAME
FILE:/tmp/krb5cc_263726_wQCVknbH7b
```

- All child processes of login process get access to kerberos through that environment variable

What happens when systemd --user runs?

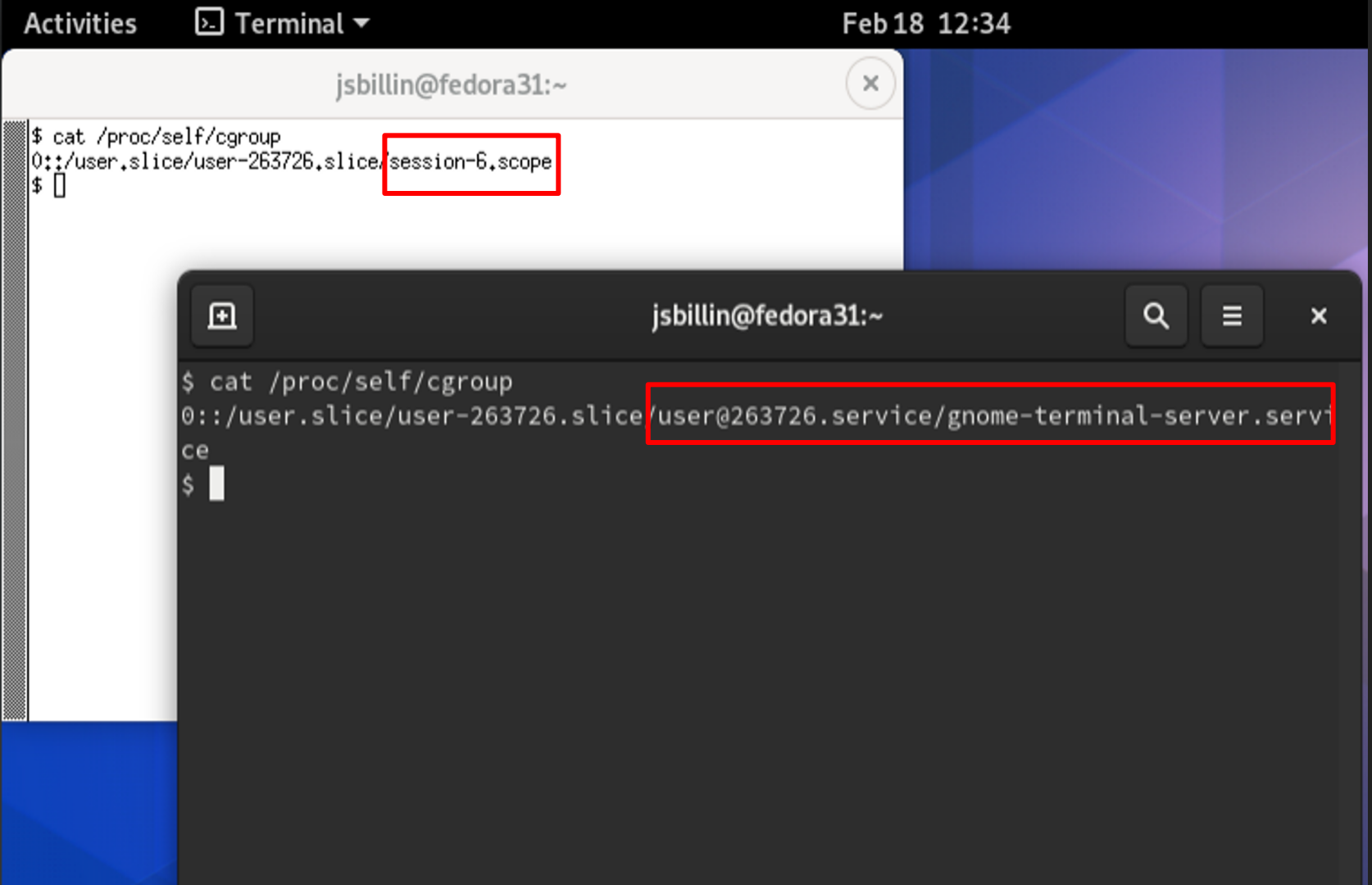
- Doesn’t have kerberos tickets
- Can’t read $HOME/.config/systemd
- Can’t write to $HOME
- No systemd --user service in $HOME/.config/systemd/user is read
- Any systemd --user service that requires access to $HOME will get read or write errors

When systemd --user runs with GNOME?

- Xorg crashes, can’t write to $HOME/.local/share/xorg/Xorg.pid-####.log
- Can’t read Dconf settings
- Nautilus crashes

What happens when systemd --user runs?

- Many graphical components are started by systemd --user
- Example: GNOME Terminal
  - gnome-terminal-server.service
  - All terminals are children of systemd --user and not login process
  - All terminals launched have no krb5 tickets


Activities
Terminal
Feb 18 12:34
jsbillin@fedora31:~
$ cat /proc/self/cgroup
0::/user.slice/user-263726.slice/session-6.scope
$
jsbillin@fedora31:~
$ cat /proc/self/cgroup
0::/user.slice/user-263726.slice/user@263726.service/gnome-terminal-server.servi
ce
$

Solutions

- Use a krb5 credential cache that is predictable
- As part of login process, run:

  `systemctl --user import-environment KRB5CCNAME`
- Since systemd --user starts *before* ccache is initialized, you'll need to run:

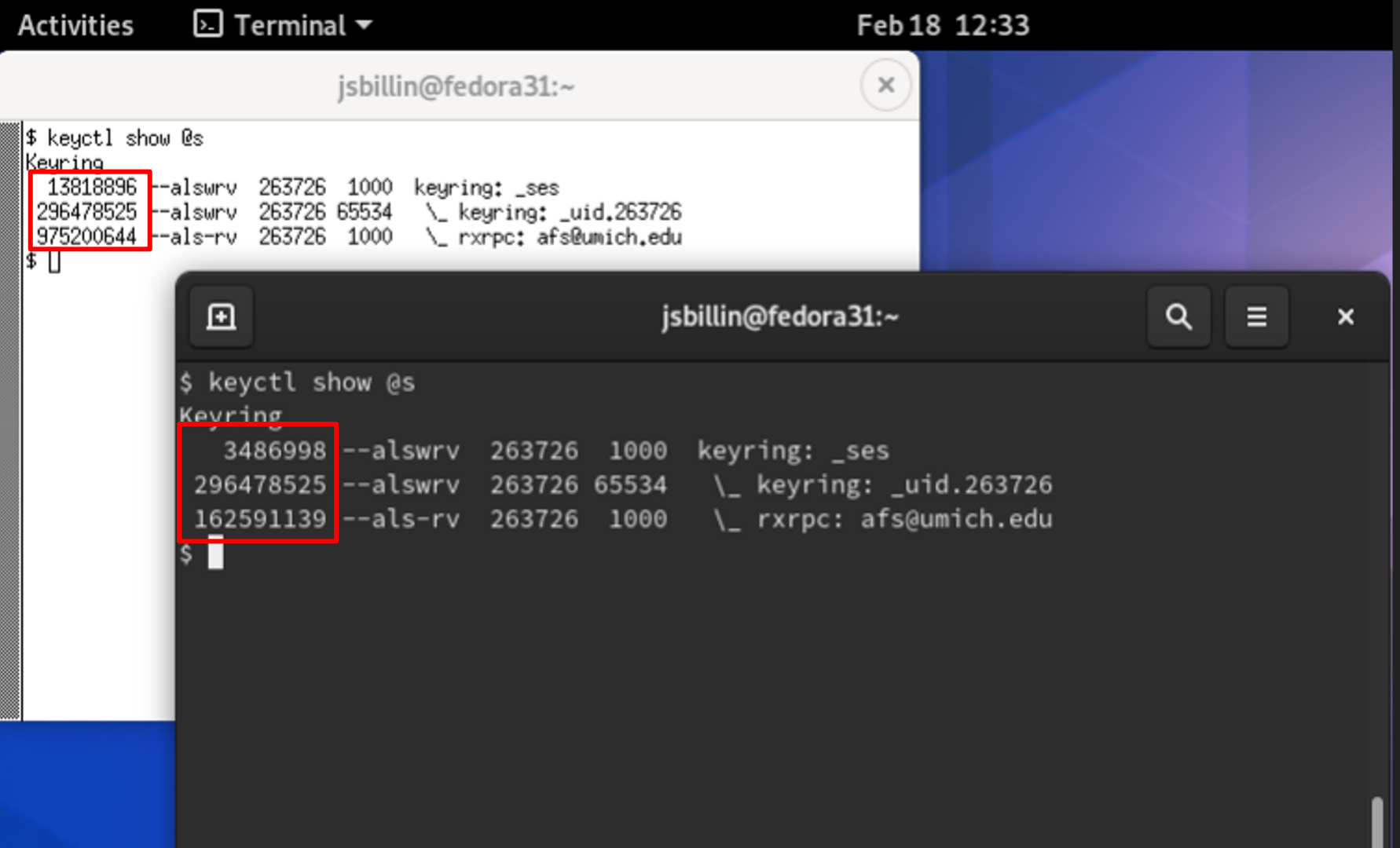
  `systemctl --user daemon-reload`
- Any failed user services also need to be restarted

Drawbacks

- Kerberos caches have traditionally been considered per-session and not per-user.
- The systemd --user processes can persist after logout
  - Safe practice is to ‘kdestroy’ at logout
  - systemd --user session is unable to touch $HOME anymore
- Multiple logins (ssh to workstation) will break other session since there is only one systemd --user session

kAFS vs. systemd

- With AFS (OpenAFS and kAFS) you need to get an AFS token from your krb5 ticket
- The systemd --user process doesn’t have either
- In addition to importing KRB5CCNAME, you need to run an ‘aklog’ to get AFS tokens
- A kdestroy does not delete tokens


Activities
Terminal
Feb 18 12:33
jsbillin@fedora31:~
$ keyctl show @s
Keyring
13818896 --alswrv 263726 1000 keyring: _ses
296478525 --alswrv 263726 65534 _ keyring: _uid.263726
975200644 --als-rv 263726 1000 _ rxrpc: afs@umich.edu
$
jsbillin@fedora31:~
$ keyctl show @s
Keyring
3486998 --alswrv 263726 1000 keyring: _ses
296478525 --alswrv 263726 65534 _ keyring: _uid.263726
162591139 --als-rv 263726 1000 _ rxrpc: afs@umich.edu
$

kAFS vs. systemd

- systemd --user session with AFS tokens will persist logouts
- Second GNOME login will discover AFS tokens from previous session in persisted systemd --user session

Solutions

- Run an aklog as part of systemd --user session

```
[Unit]
Description=Set up AFS tokens
Before=dbus.socket
Requires=afs.mount
[Service]
Type=simple
Environment=KRB5CCNAME=KEYRING:persistent:%U
RemainAfterExit=yes
ExecStart=/usr/bin/aklog
StandardOutput=syslog
[Install]
WantedBy=default.target
```

Drawbacks

- AFS tokens still persist with systemd --user
- Renewing tokens in login session does not renew tokens in systemd --user session
  - Need to restart the aklog.service to renew tokens
- systemd --user does not handle krb5 ticket and AFS token expirations gracefully
- Usually just crashes GNOME

Best solution

- Disable systemd --user entirely
  `systemctl mask user@.service`
- Disabling user systemd breaks some software
  - Flatpak
  - Gnome tracker
- In latest GNOME, disabling systemd --user breaks GNOME desktop and GDM, must use other Display Manager and Desktop Environment
  - I use lightdm and MATE

Why use kAFS?

If you’re at an institution that uses AFS

- kAFS is more efficient than OpenAFS
  - Uses kernel VFS layer
  - Less overhead from network stack
  - Not limited by cache manager user space
  - Write latency is lower
- Officially part of Linux kernel and uses cool new features
- More open license, no 3rd party kmod

What needs to be implemented

- Fix systemd issues
- More AFS tools
  - Need ‘fs’ command for ACL manipulation
  - ‘vos’, ‘pts’, ‘bos’ already written but need testing
  - Use “user” keyring instead of “session” keyring for storing AFS token
- More distros should turn on kAFS and rxrpc kernel modules

Planned features

- inotify support
- SELinux labels and other extended attributes
- Container namespacing

Thanks!

David Howells (Red Hat)

Jeff Altman (Auristor)

OpenAFS Foundation

Questions?

Contact me:
jsbillings@jsbillings.org