LINUX PLUMBERS CONFERENCE 2021
SEPTEMBER 20-24, 2021

On-demand EventFS
to reduce Linux Tracer memory footprint

Steven Rostedt (srostedt@vmware.com)
Ajay Kaher (akaher@vmware.com)

Agenda

- Brief Introduction to Linux Tracer Events, Multiple instances of Tracer/Events

- Linux Tracer Memory Footprint

- On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Code snippet of Eventfs APIs/Structure

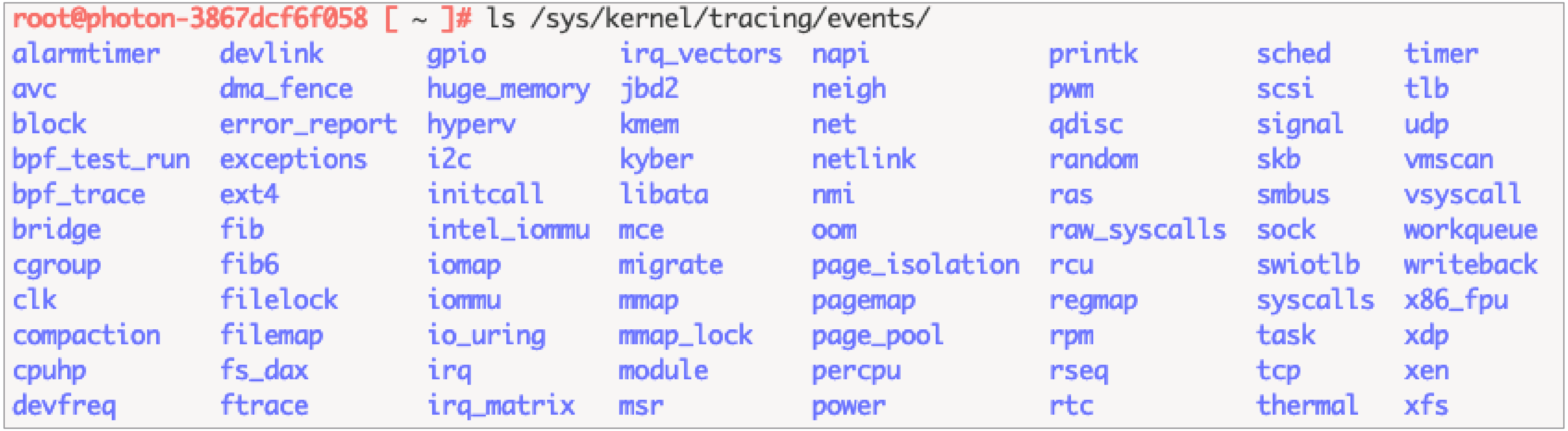
- Conclusion

- On going task, Suggestions/Feedback

Linux Tracing and Events hierarchy:

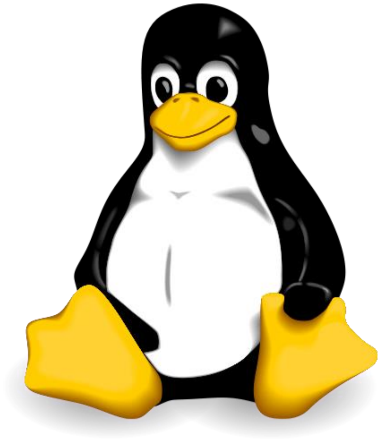
Linux Tracer is used for debugging and performance analysis of Kernel

Events Tracing Infrastructure:

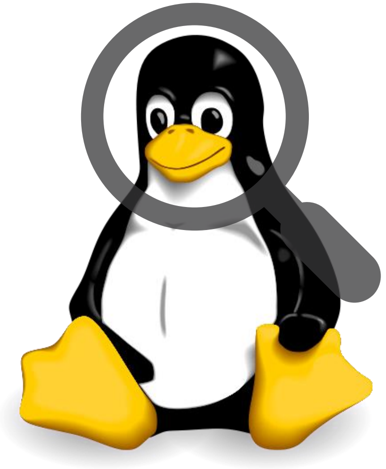


```
root@photon-3867dcf6f058 [ ~ ]# ls /sys/kernel/tracing/events/
alarmtimer    devlink       gpio         irq_vectors  napi            printk        sched     timer
avc           dma_fence     huge_memory  jbd2         neigh           pwm           scsi      tlb
block         error_report  hyperv       kmem         net             qdisc         signal    udp
bpf_test_run  exceptions    i2c          kyber        netlink         random        skb       vmscan
bpf_trace     ext4          initcall     libata       nmi             ras           smbus     vsyscall
bridge        fib           intel_iommu  mce          oom             raw_syscalls  sock      workqueue
cgroup        fib6          iomap        migrate      page_isolation  rcu           swiotlb   writeback
clk           filelock      iommu        mmap         pagemap         regmap        syscalls  x86_fpu
compaction    filemap       io_uring     mmap_lock    page_pool       rpm           task      xdp
cpuhp         fs_dax        irq          module       percpu          rseq          tcp       xen
devfreq       ftrace        irq_matrix   msr          power           rtc           thermal   xfs
```

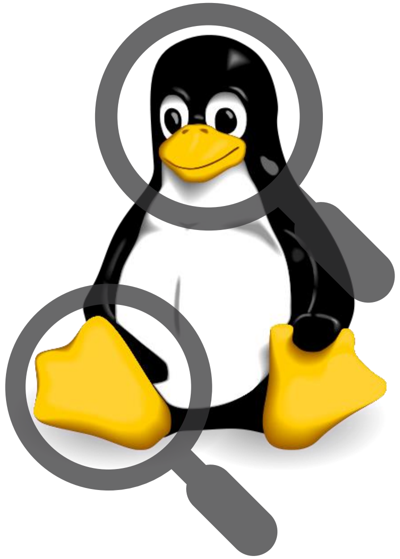
Multiple Instances of Tracing and Events



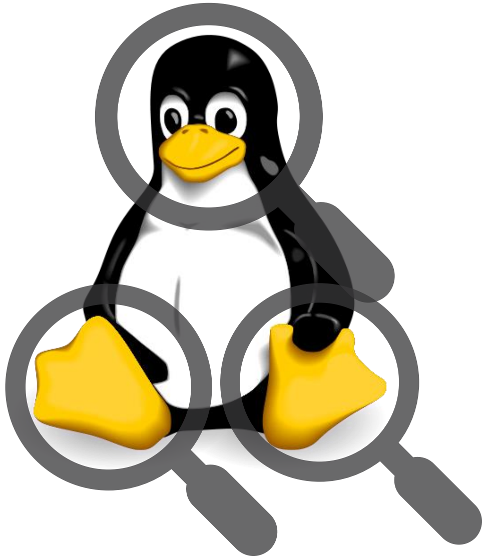
Multiple Instances of Tracing and Events



Multiple Instances of Tracing and Events

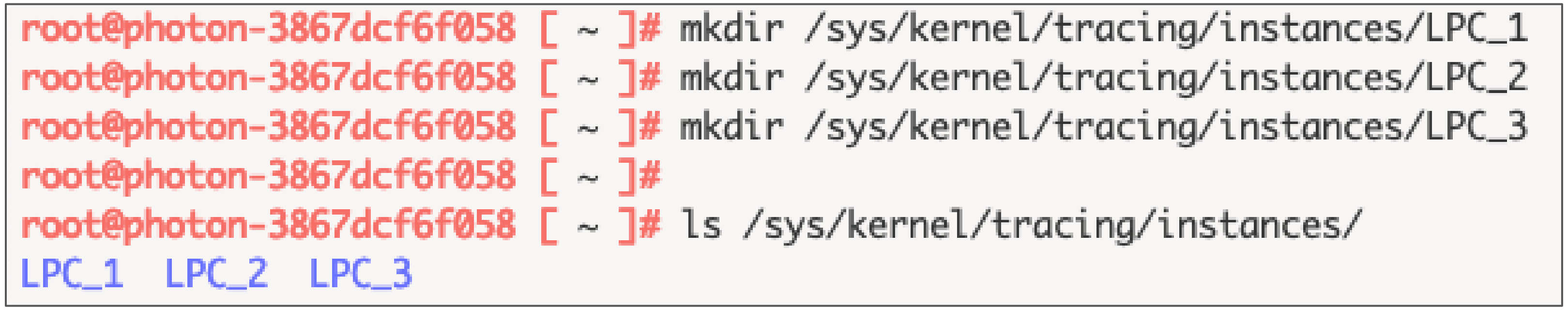


Multiple Instances of Tracing and Events

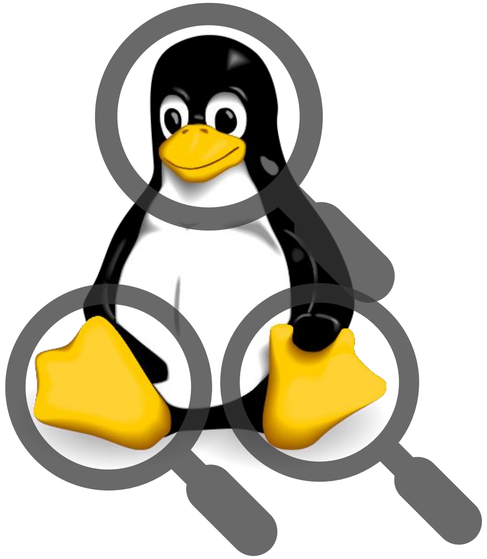


Multiple Instances of Tracing and Events

Linux tracer provides mechanism to have multiple instances of 'tracing'

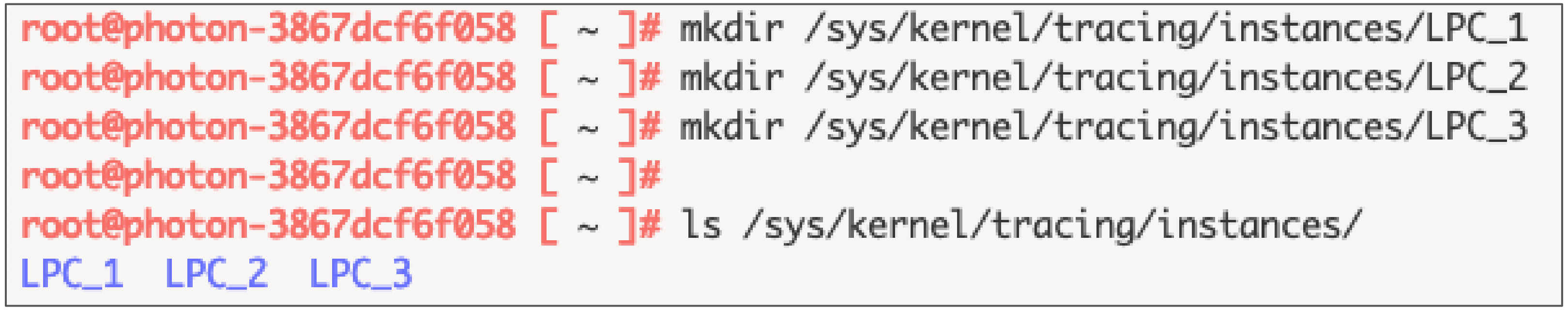


```
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_1
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_2
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_3
root@photon-3867dcf6f058 [ ~ ]#
root@photon-3867dcf6f058 [ ~ ]# ls /sys/kernel/tracing/instances/
LPC_1  LPC_2  LPC_3
```

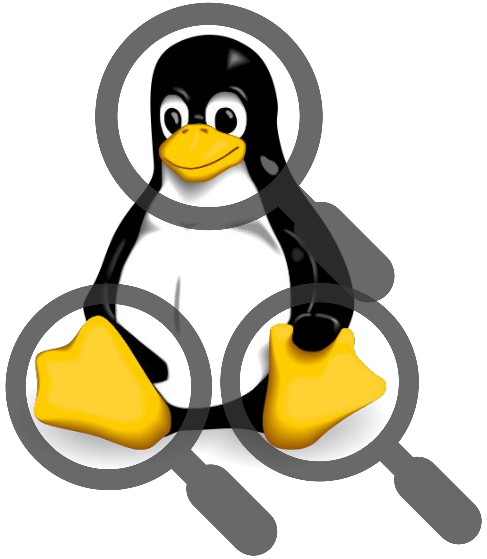


Multiple Instances of Tracing and Events

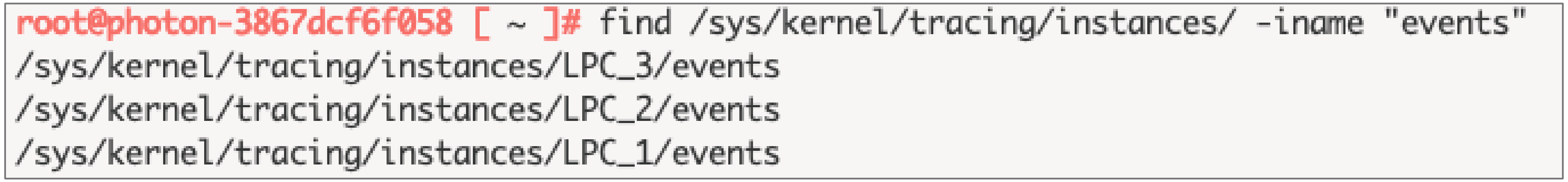
Linux tracer provides mechanism to have multiple instances of 'tracing'



```
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_1
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_2
root@photon-3867dcf6f058 [ ~ ]# mkdir /sys/kernel/tracing/instances/LPC_3
root@photon-3867dcf6f058 [ ~ ]#
root@photon-3867dcf6f058 [ ~ ]# ls /sys/kernel/tracing/instances/
LPC_1  LPC_2  LPC_3
```



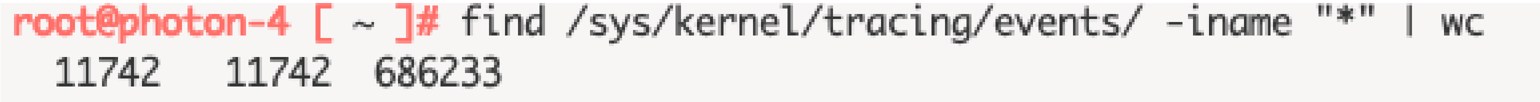
Each tracing instance have individual events directory known as 'Events Tracing Infrastructure'



```
root@photon-3867dcf6f058 [ ~ ]# find /sys/kernel/tracing/instances/ -iname "events"
/sys/kernel/tracing/instances/LPC_3/events
/sys/kernel/tracing/instances/LPC_2/events
/sys/kernel/tracing/instances/LPC_1/events
```

Problem Statement: Linux Tracer Memory Footprint

- 'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)

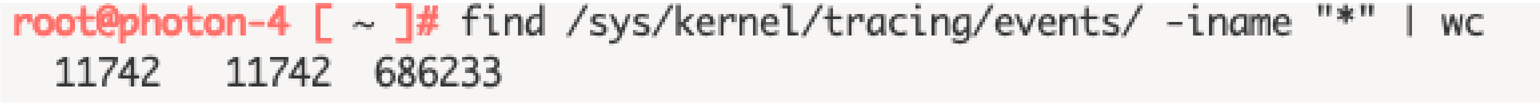


```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- 'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB

Problem Statement: Linux Tracer Memory Footprint

- **'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)**



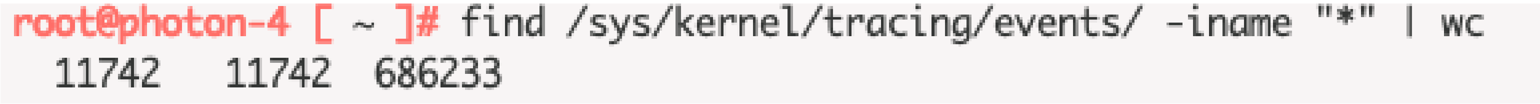
```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- **'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB**

Linux Tracer

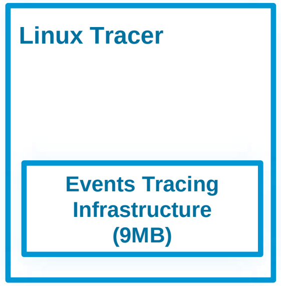
Problem Statement: Linux Tracer Memory Footprint

- **'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)**



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- **'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB**

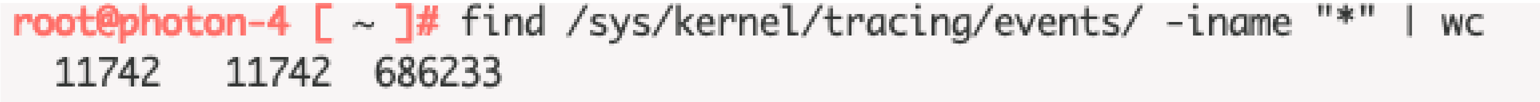


Linux Tracer

Events Tracing Infrastructure (9MB)

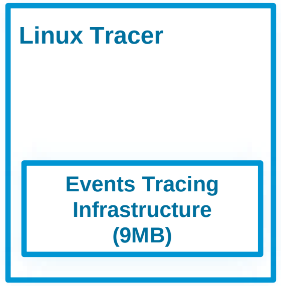
Problem Statement: Linux Tracer Memory Footprint

- 'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

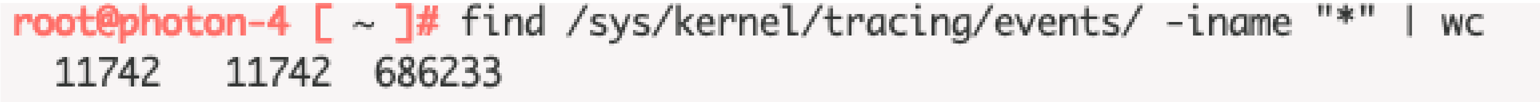
- 'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB



1st Instance

Problem Statement: Linux Tracer Memory Footprint

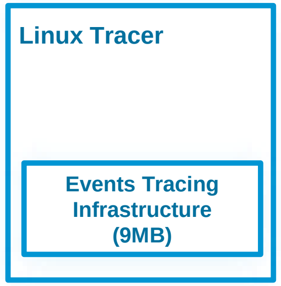
- 'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- 'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB

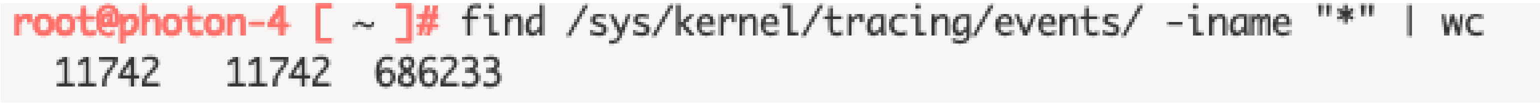
- Multiple Instances of Linux Events with-in Linux Tracer = 'No of Instances * 9 MB'.



1st Instance

Problem Statement: Linux Tracer Memory Footprint

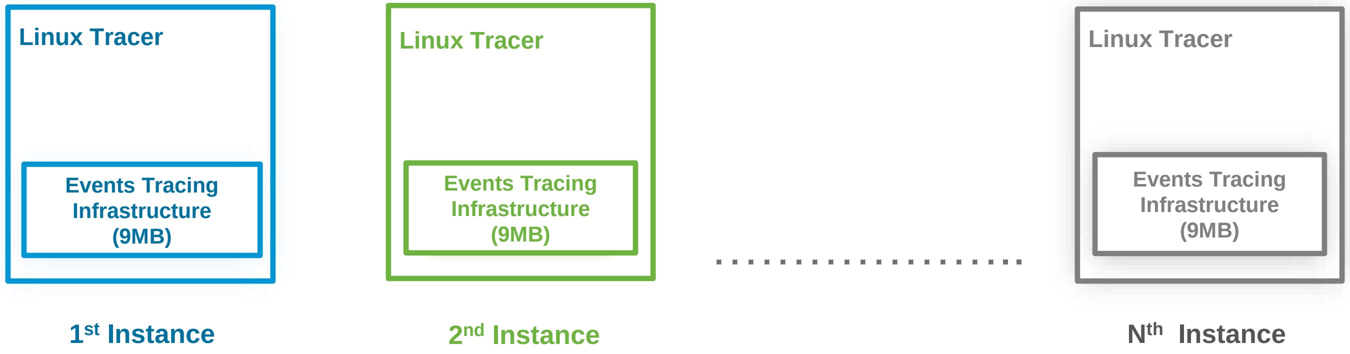
- 'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

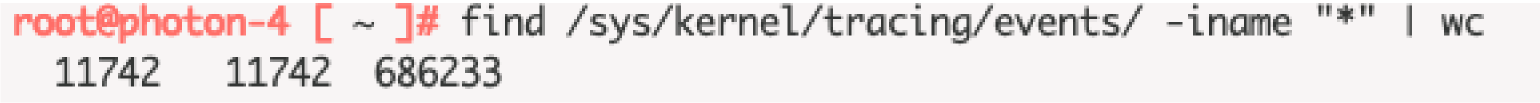
- 'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB

- Multiple Instances of Linux Events with-in Linux Tracer = ‘No of Instances * 9 MB’.


Linux Tracer
Events Tracing Infrastructure (9MB)
1st Instance
Linux Tracer
Events Tracing Infrastructure (9MB)
2nd Instance
…………………
Linux Tracer
Events Tracing Infrastructure (9MB)
Nth Instance

Problem Statement: Linux Tracer Memory Footprint

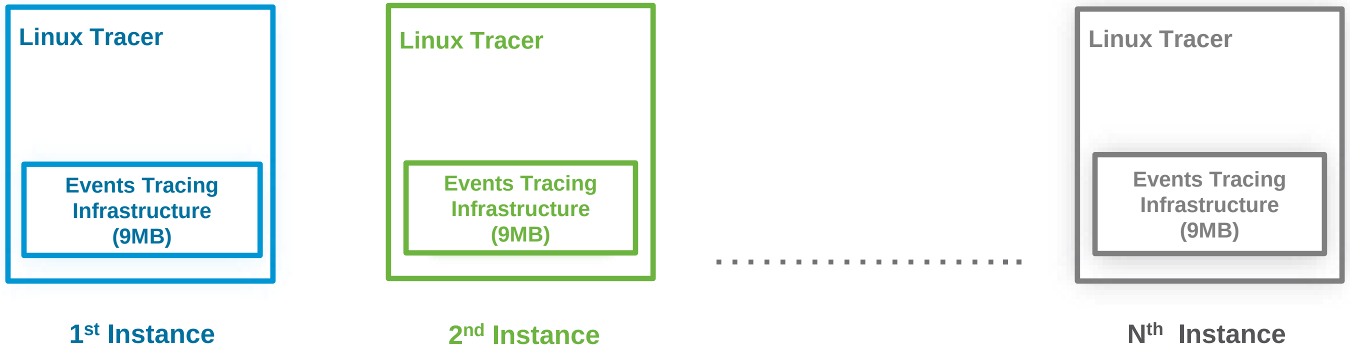
- **'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)**



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- **'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB**

- **Multiple Instances of Linux Events with-in Linux Tracer = 'No of Instances * 9 MB'.**

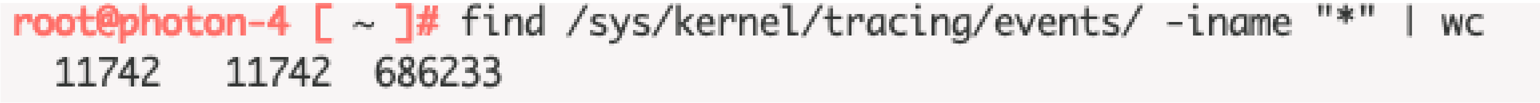


Linux Tracer
Events Tracing Infrastructure (9MB)
1st Instance
Linux Tracer
Events Tracing Infrastructure (9MB)
2nd Instance
Linux Tracer
Events Tracing Infrastructure (9MB)
Nth Instance

Goal: Reduce memory footprint of 'Events Tracing Infrastructure'

Problem Statement: Linux Tracer Memory Footprint

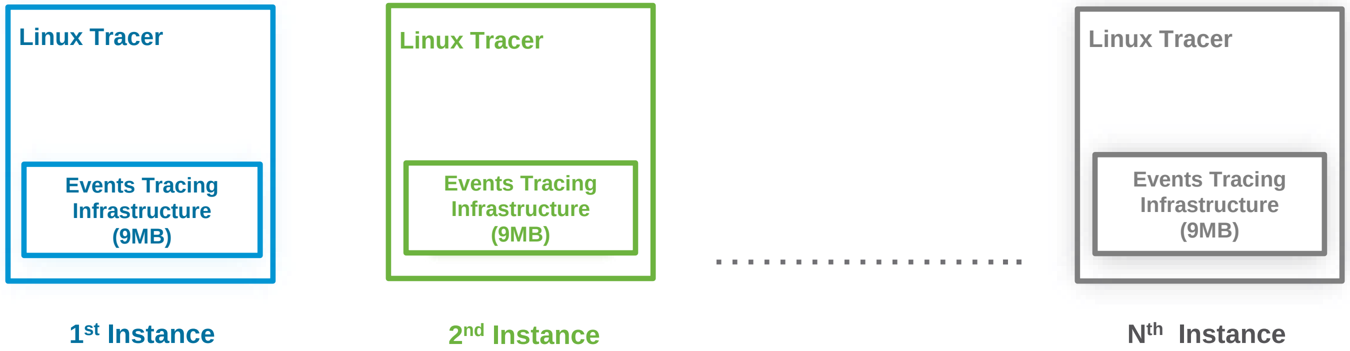
- 'Events Tracing Infrastructure' contains lot of files/directories (although depending upon the Kernel config)



```
root@photon-4 [ ~ ]# find /sys/kernel/tracing/events/ -iname "*" | wc
  11742   11742  686233
```

- 'Events Tracing Infrastructure' has 11742 files/directories = ~ 9 MB

- Multiple Instances of Linux Events with-in Linux Tracer = 'No of Instances * 9 MB'.


Linux Tracer
Events Tracing Infrastructure (9MB)
1st Instance
Linux Tracer
Events Tracing Infrastructure (9MB)
2nd Instance
…………………
Linux Tracer
Events Tracing Infrastructure (9MB)
Nth Instance

Goal: Reduce memory footprint of 'Events Tracing Infrastructure'

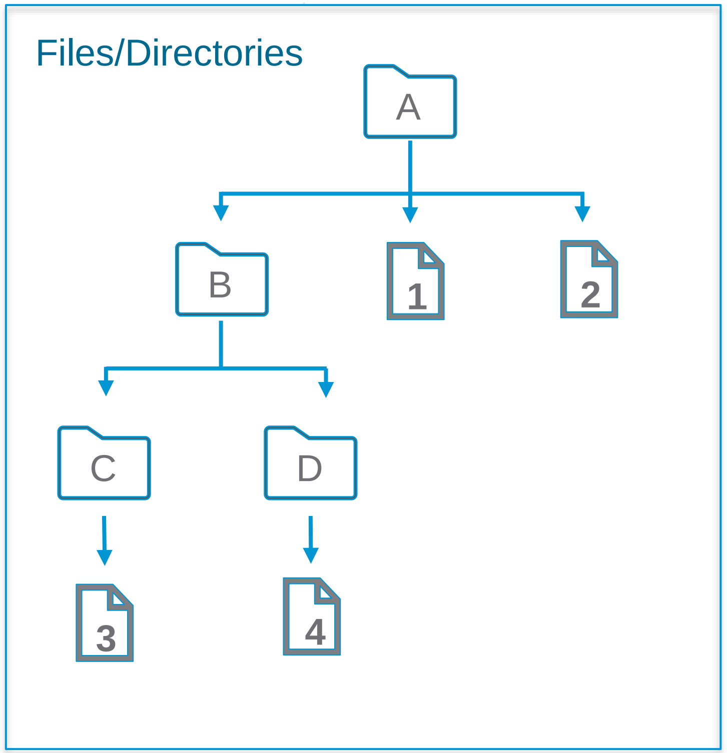
Target: 75% reduction in memory footprint of Events

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

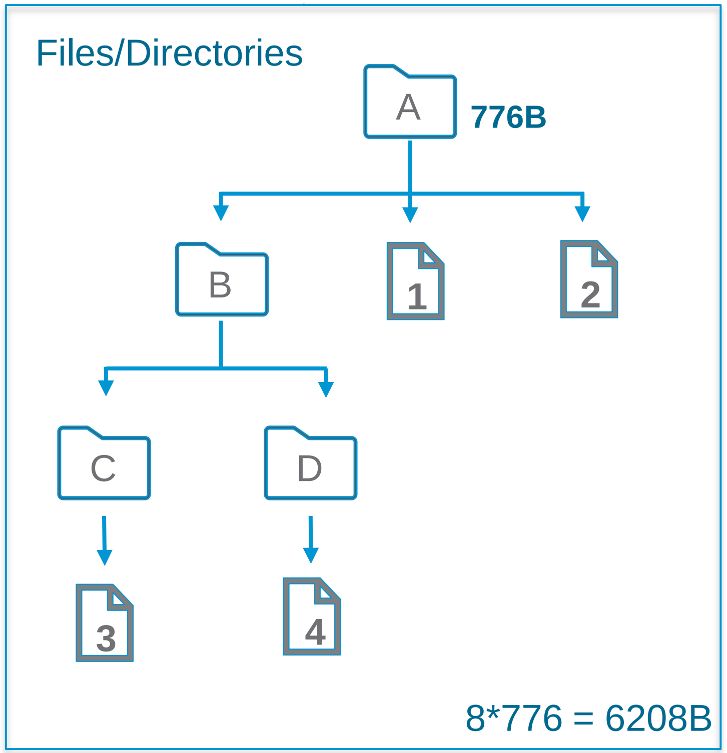
- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

Solution: On-demand Eventfs to improve 'Tracer Memory Footprint'

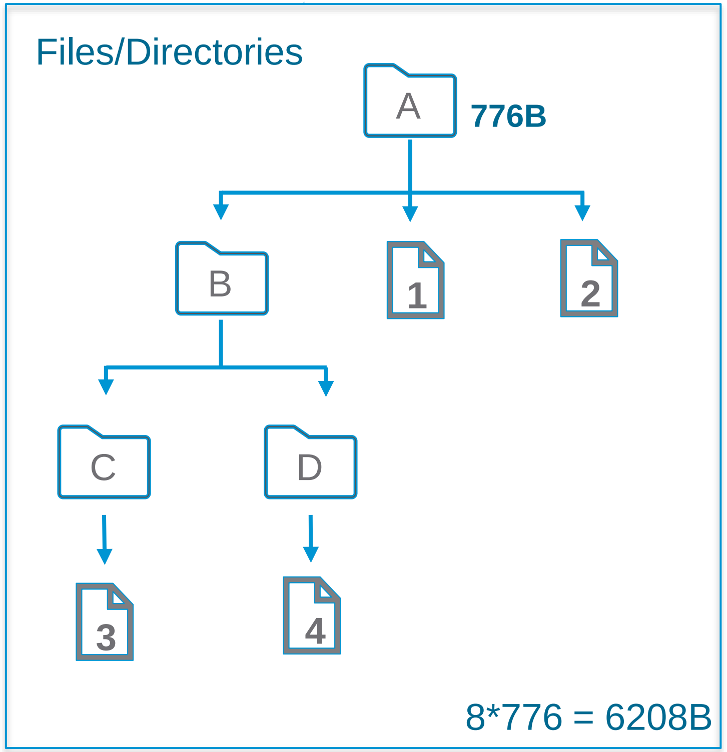
- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



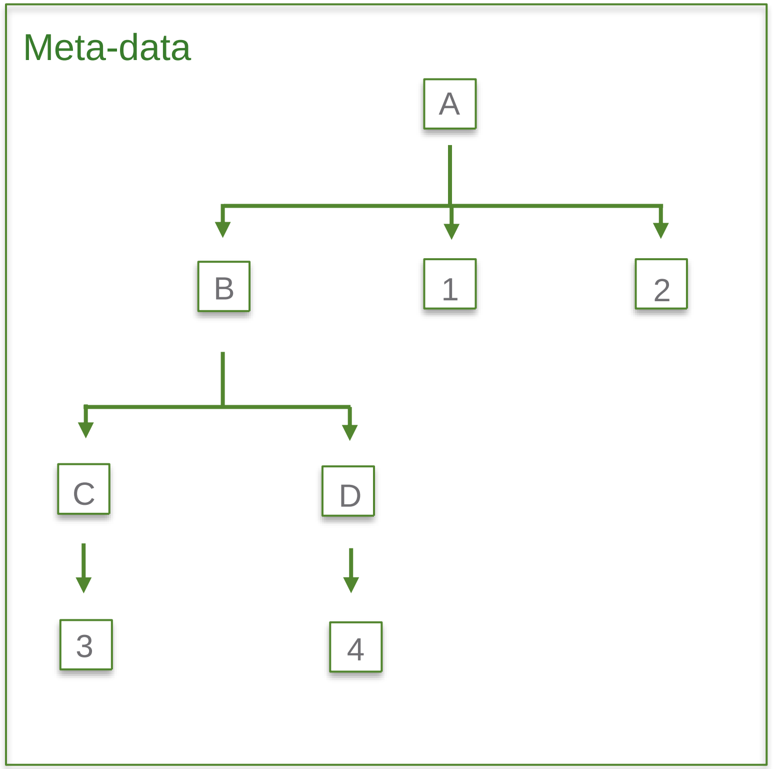
Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



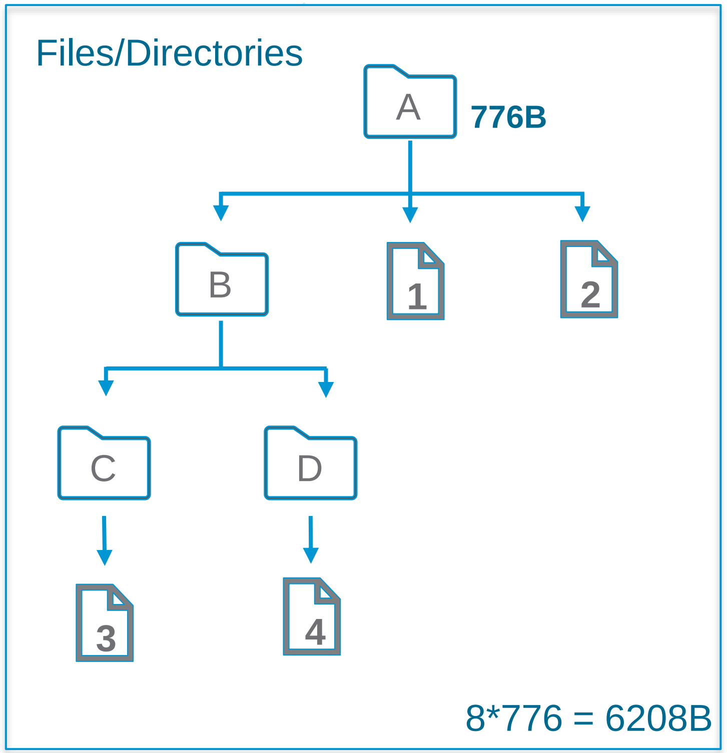
Tracefs



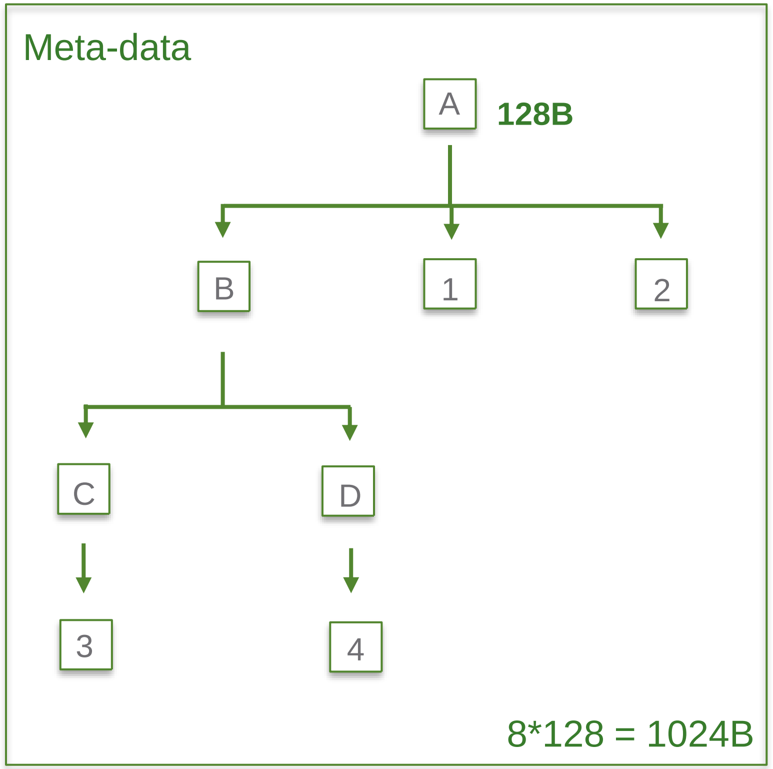
On-Demand Eventfs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



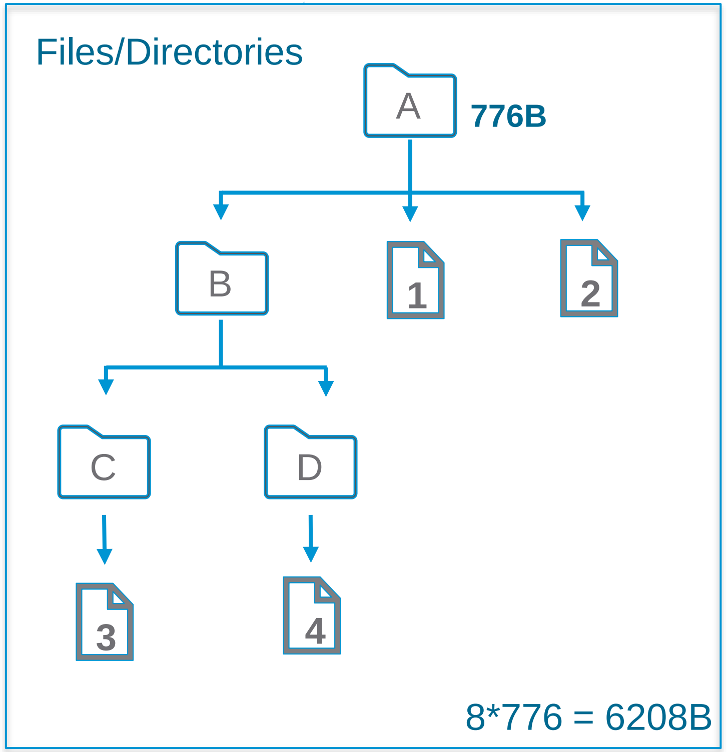
Tracefs



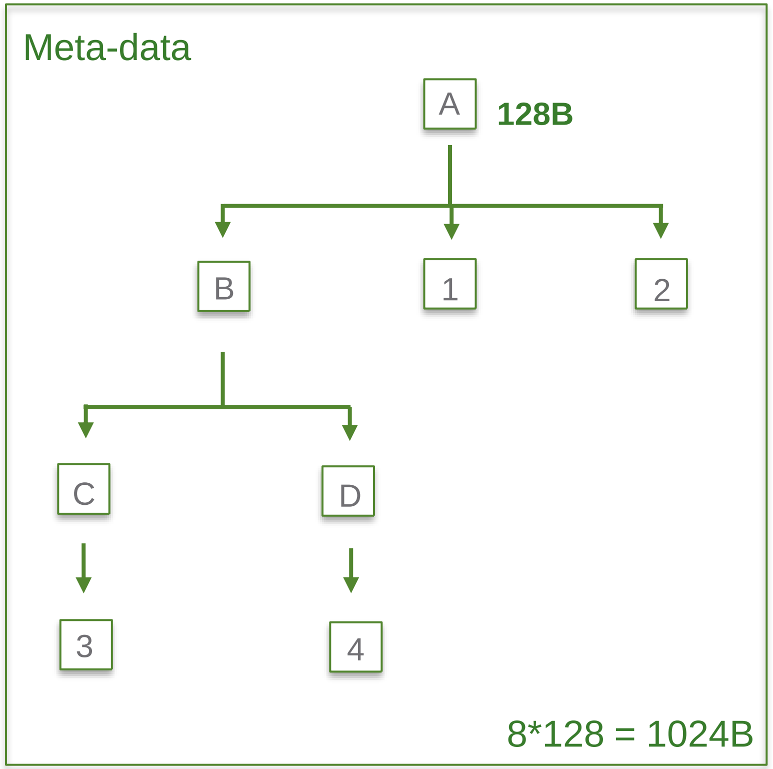
On-Demand Eventfs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

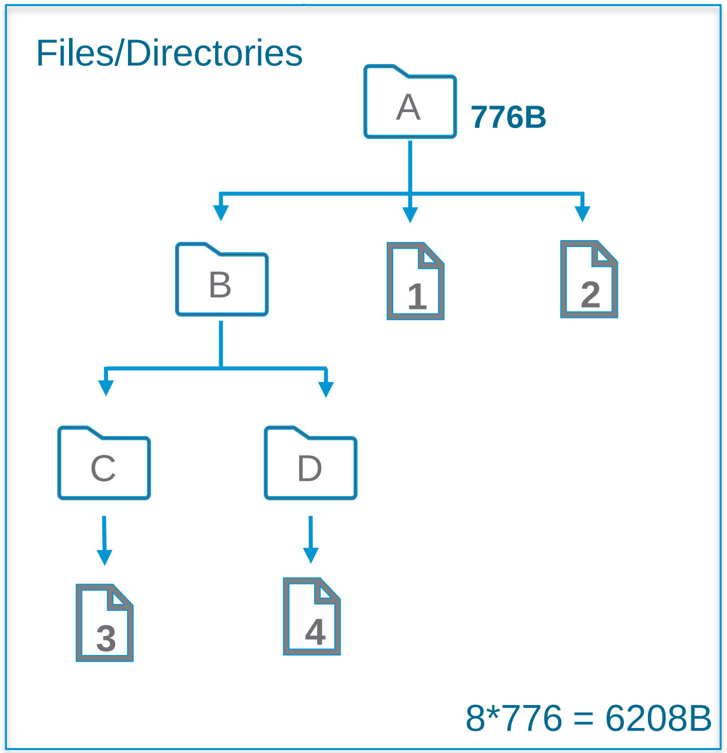


On-Demand Eventfs

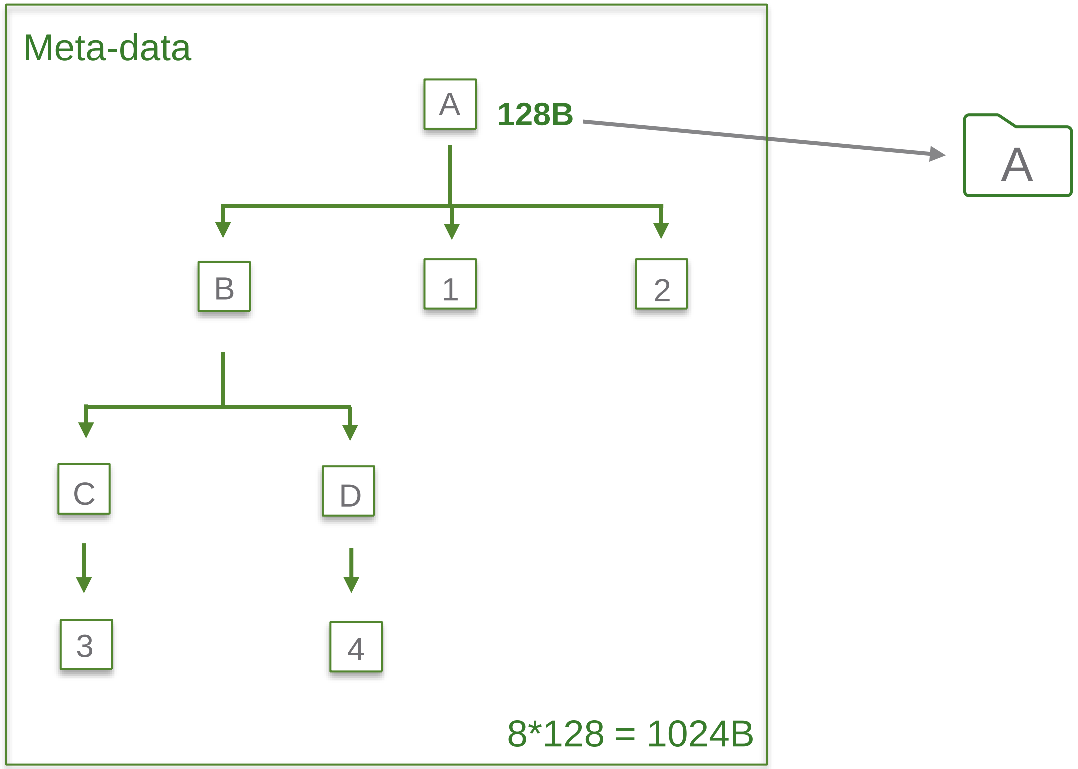
On-Demand Eventfs consumes 80% less memory as compared to Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

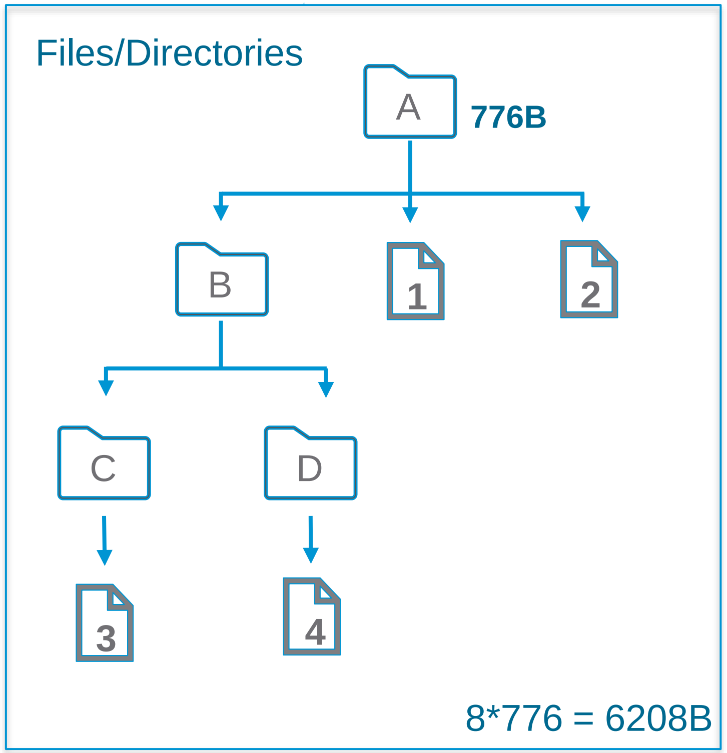


On-Demand Eventfs

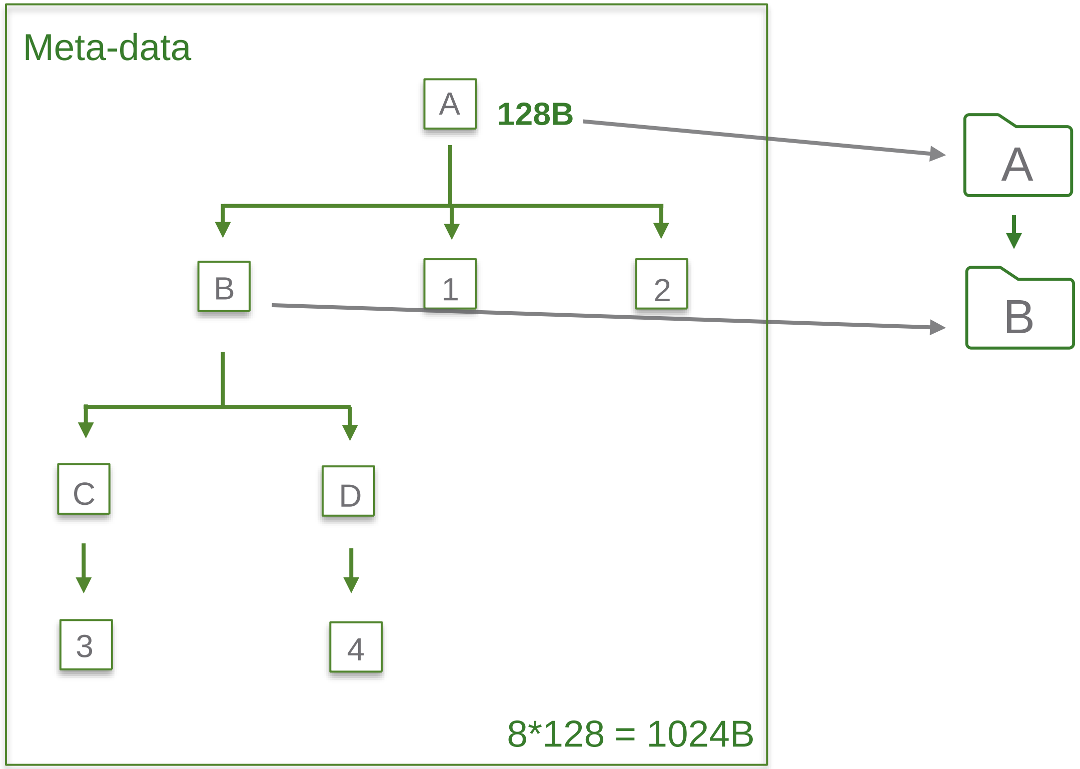
On-Demand Eventfs consumes 80% less memory as compared to Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

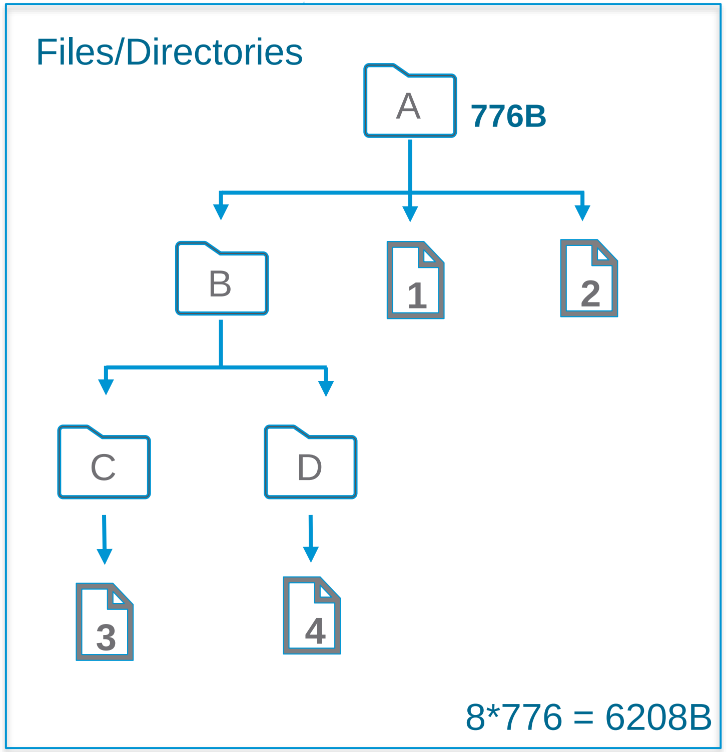


On-Demand Eventfs

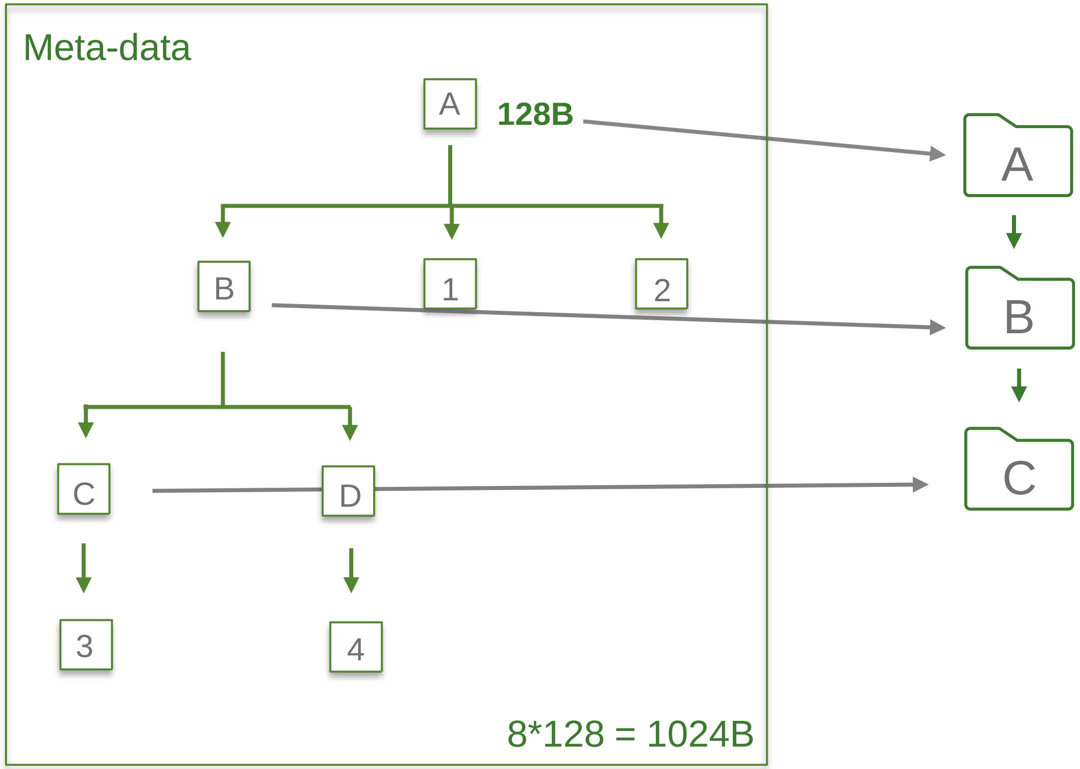
On-Demand Eventfs consumes 80% less memory as compared to Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

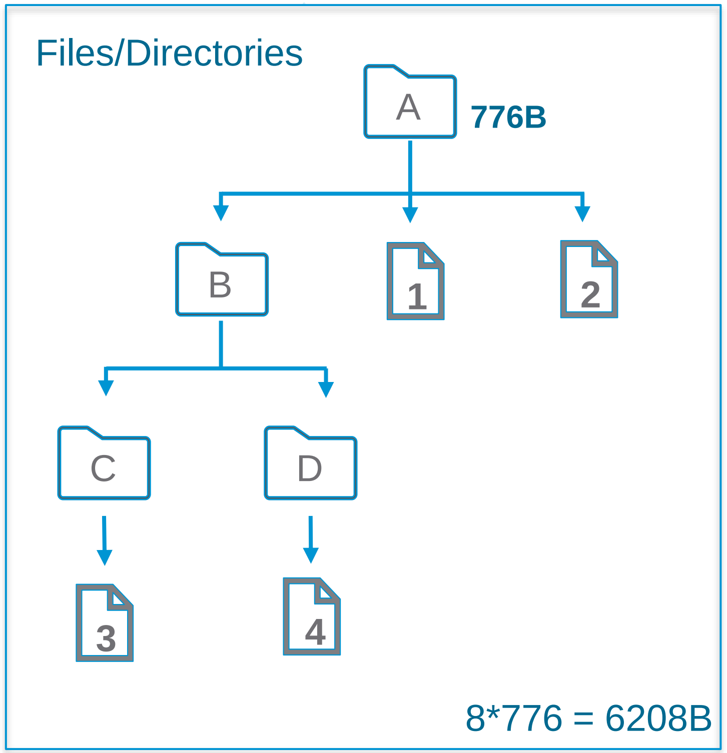


On-Demand Eventfs

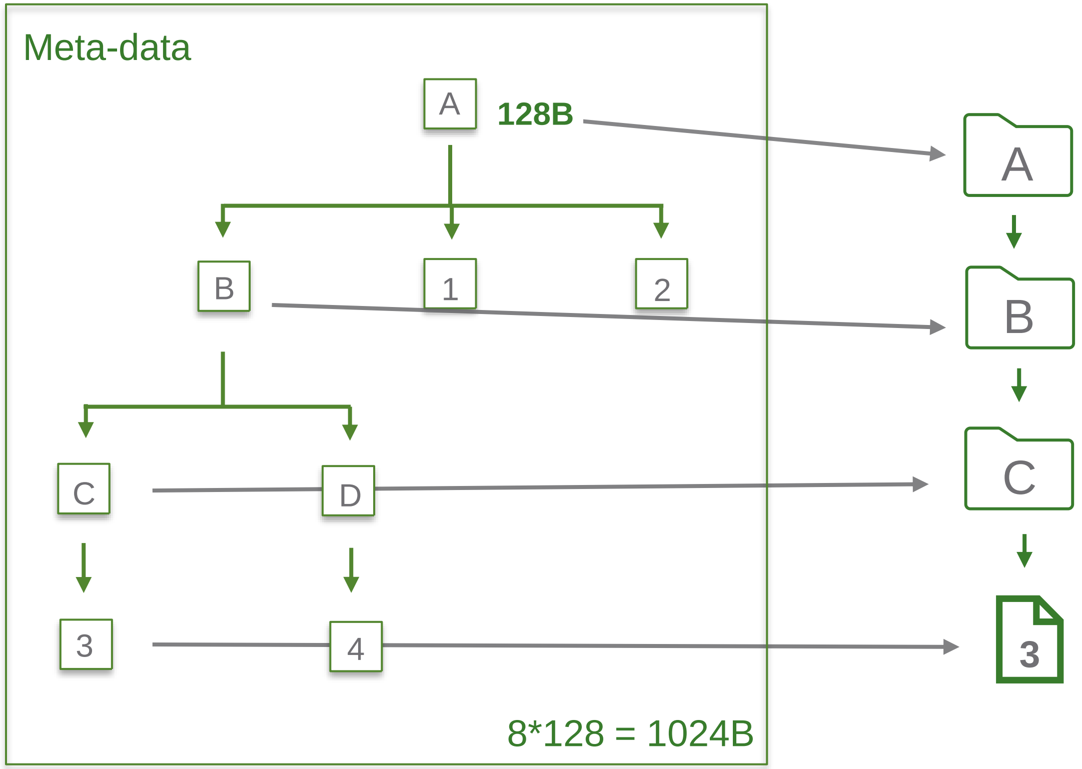
On-Demand Eventfs consumes 80% less memory as compared to Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



Tracefs

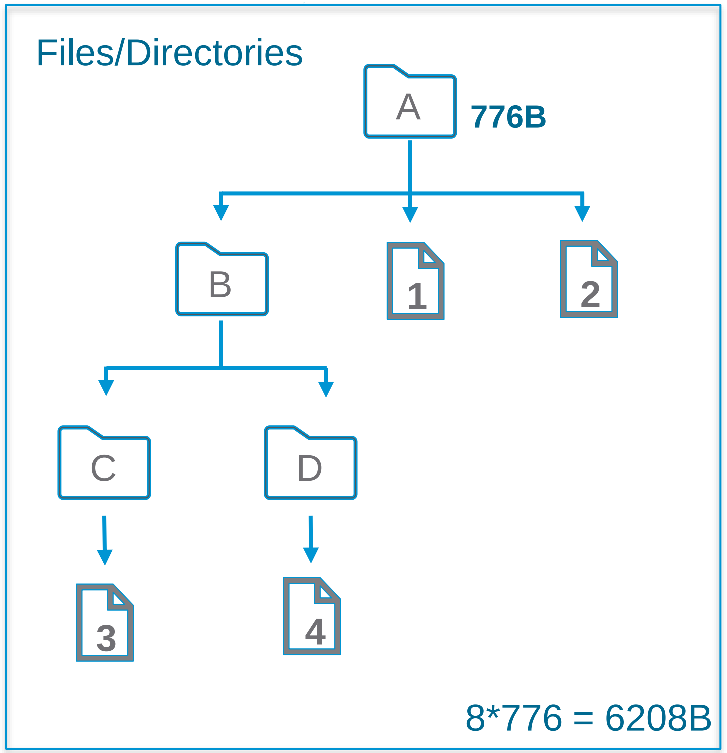


On-Demand Eventfs

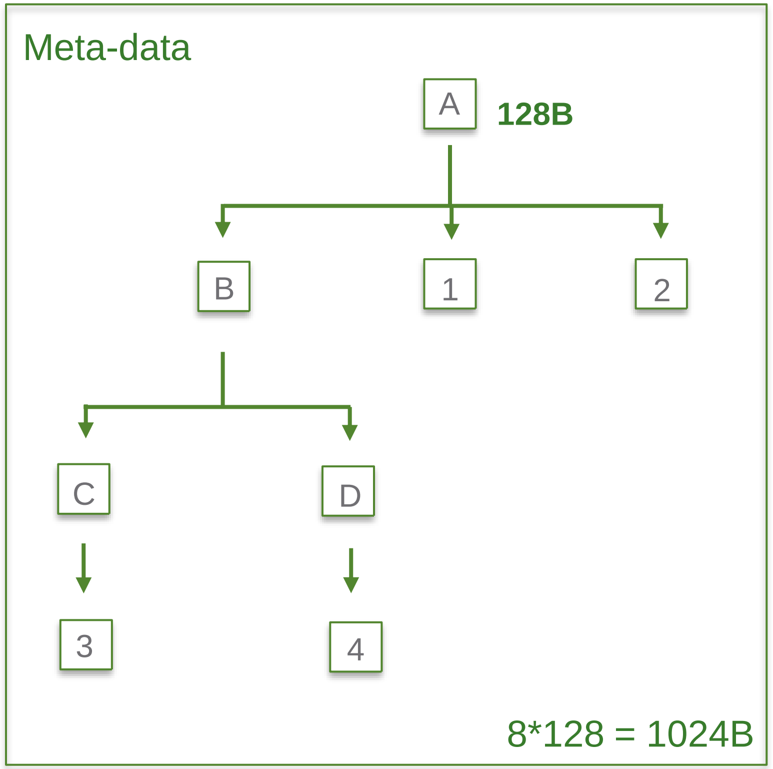
On-Demand Eventfs consumes 80% less memory as compared to Tracefs

Solution: On-demand Eventfs to improve ‘Tracer Memory Footprint’

- Instead of inode and dentry structure, just keep the meta-data of files/directories.
- On-demand use the meta-data to create the files/directories and delete them if no longer requires.



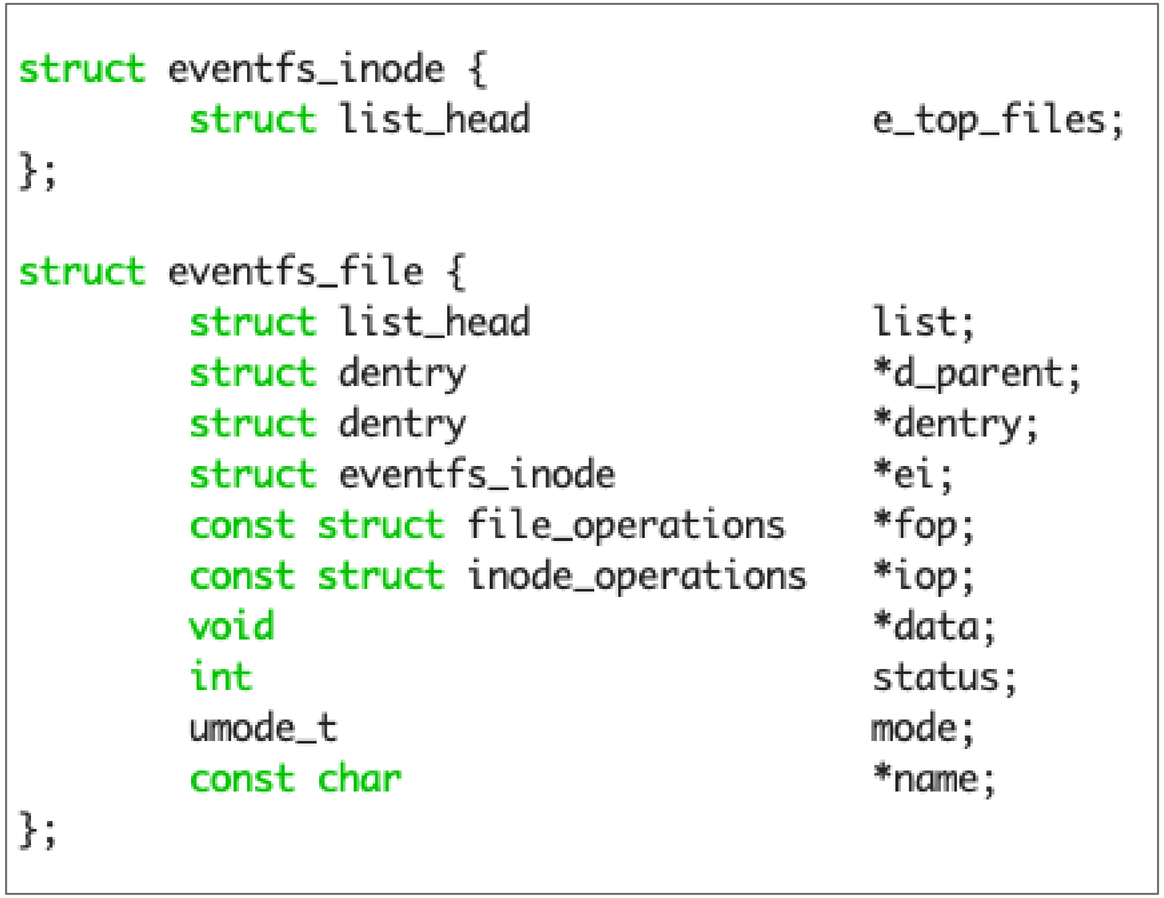
Tracefs



On-Demand Eventfs

On-Demand Eventfs consumes 80% less memory as compared to Tracefs

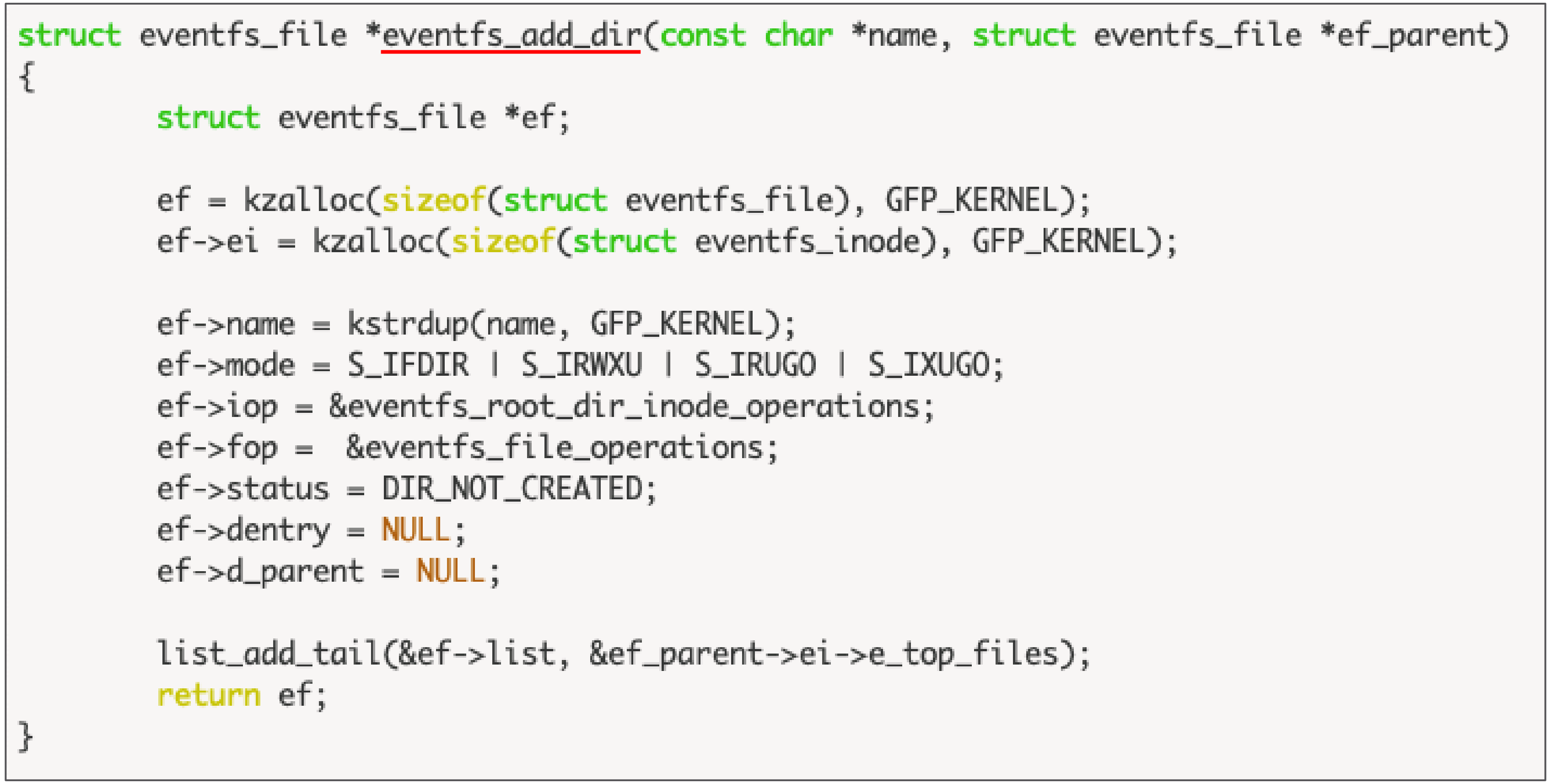
Eventfs: Metadata Structures - eventfs_inode, eventfs_file



```
struct eventfs_inode {
        struct list_head                 e_top_files;
};

struct eventfs_file {
        struct list_head                 list;
        struct dentry                    *d_parent;
        struct dentry                    *dentry;
        struct eventfs_inode             *ei;
        const struct file_operations     *fop;
        const struct inode_operations    *iop;
        void                             *data;
        int                              status;
        umode_t                          mode;
        const char                       *name;
};
```

Eventfs: Add files and directories



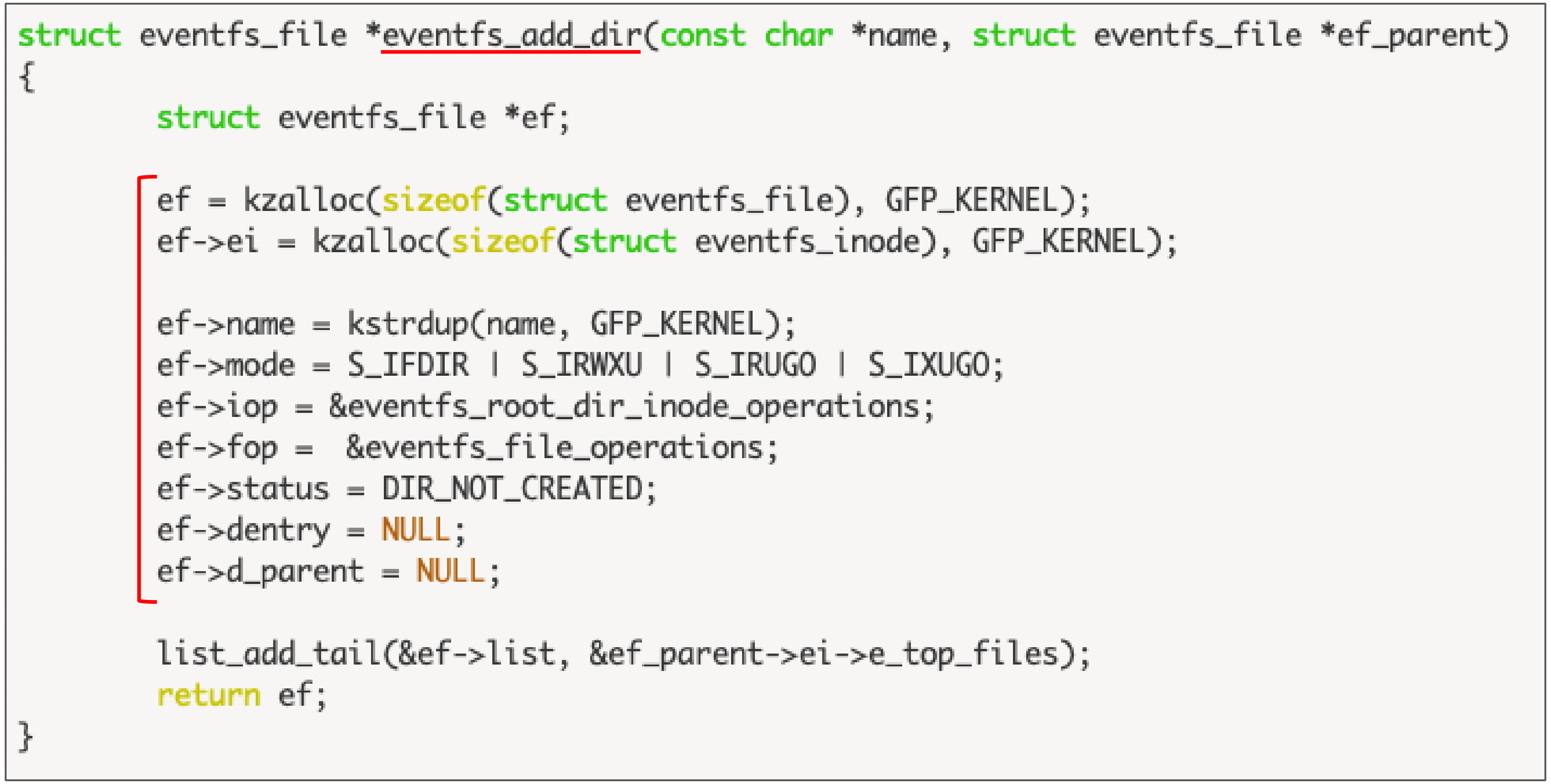
```
struct eventfs_file *eventfs_add_dir(const char *name, struct eventfs_file *ef_parent)
{
        struct eventfs_file *ef;

        ef = kzalloc(sizeof(struct eventfs_file), GFP_KERNEL);
        ef->ei = kzalloc(sizeof(struct eventfs_inode), GFP_KERNEL);

        ef->name = kstrdup(name, GFP_KERNEL);
        ef->mode = S_IFDIR | S_IRWXU | S_IRUGO | S_IXUGO;
        ef->iop = &eventfs_root_dir_inode_operations;
        ef->fop =  &eventfs_file_operations;
        ef->status = DIR_NOT_CREATED;
        ef->dentry = NULL;
        ef->d_parent = NULL;

        list_add_tail(&ef->list, &ef_parent->ei->e_top_files);
        return ef;
}
```

Eventfs: Add files and directories



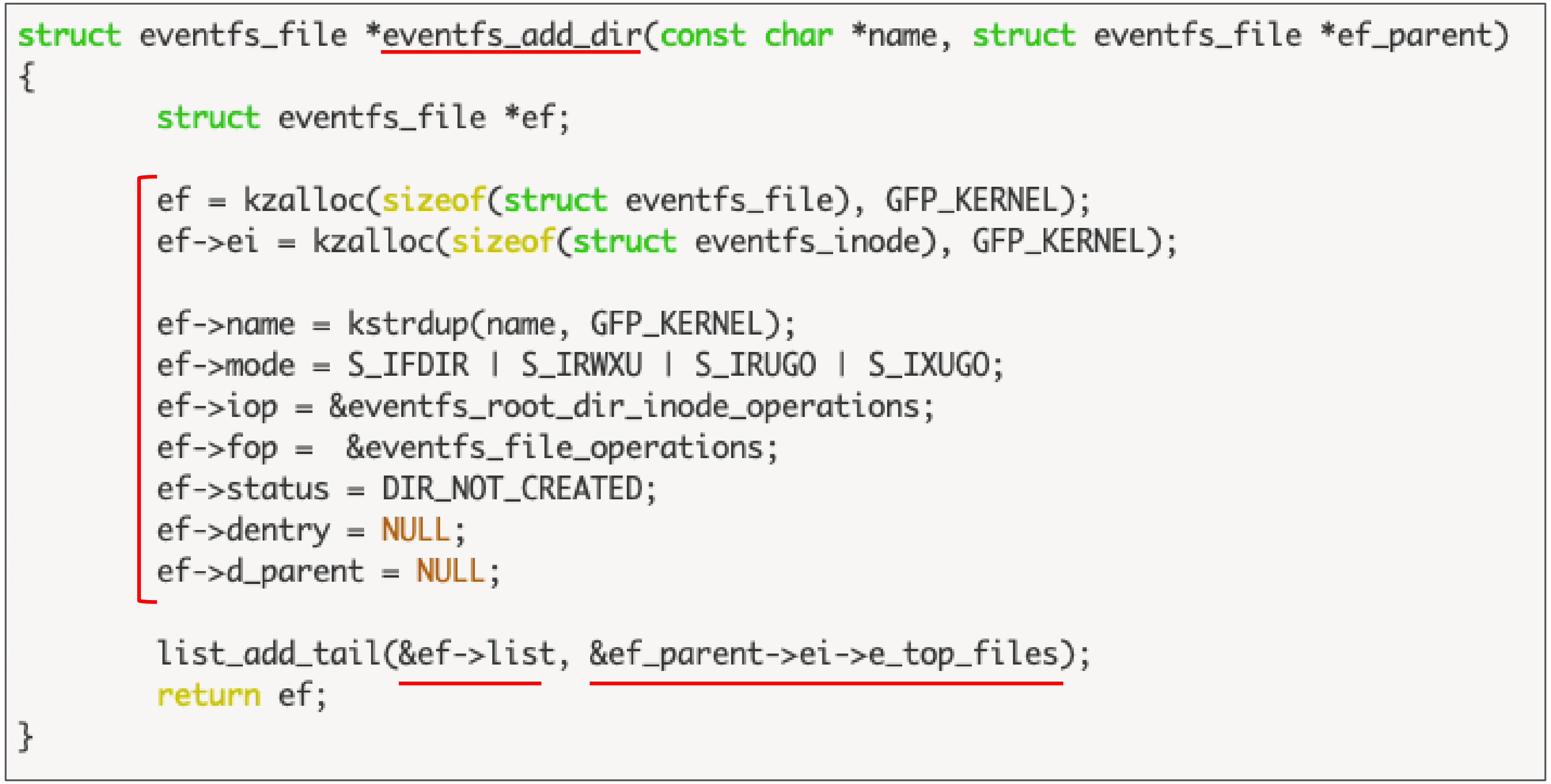
```
struct eventfs_file *eventfs_add_dir(const char *name, struct eventfs_file *ef_parent)
{
        struct eventfs_file *ef;

        ef = kzalloc(sizeof(struct eventfs_file), GFP_KERNEL);
        ef->ei = kzalloc(sizeof(struct eventfs_inode), GFP_KERNEL);

        ef->name = kstrdup(name, GFP_KERNEL);
        ef->mode = S_IFDIR | S_IRWXU | S_IRUGO | S_IXUGO;
        ef->iop = &eventfs_root_dir_inode_operations;
        ef->fop =  &eventfs_file_operations;
        ef->status = DIR_NOT_CREATED;
        ef->dentry = NULL;
        ef->d_parent = NULL;

        list_add_tail(&ef->list, &ef_parent->ei->e_top_files);
        return ef;
}
```

Eventfs: Add files and directories



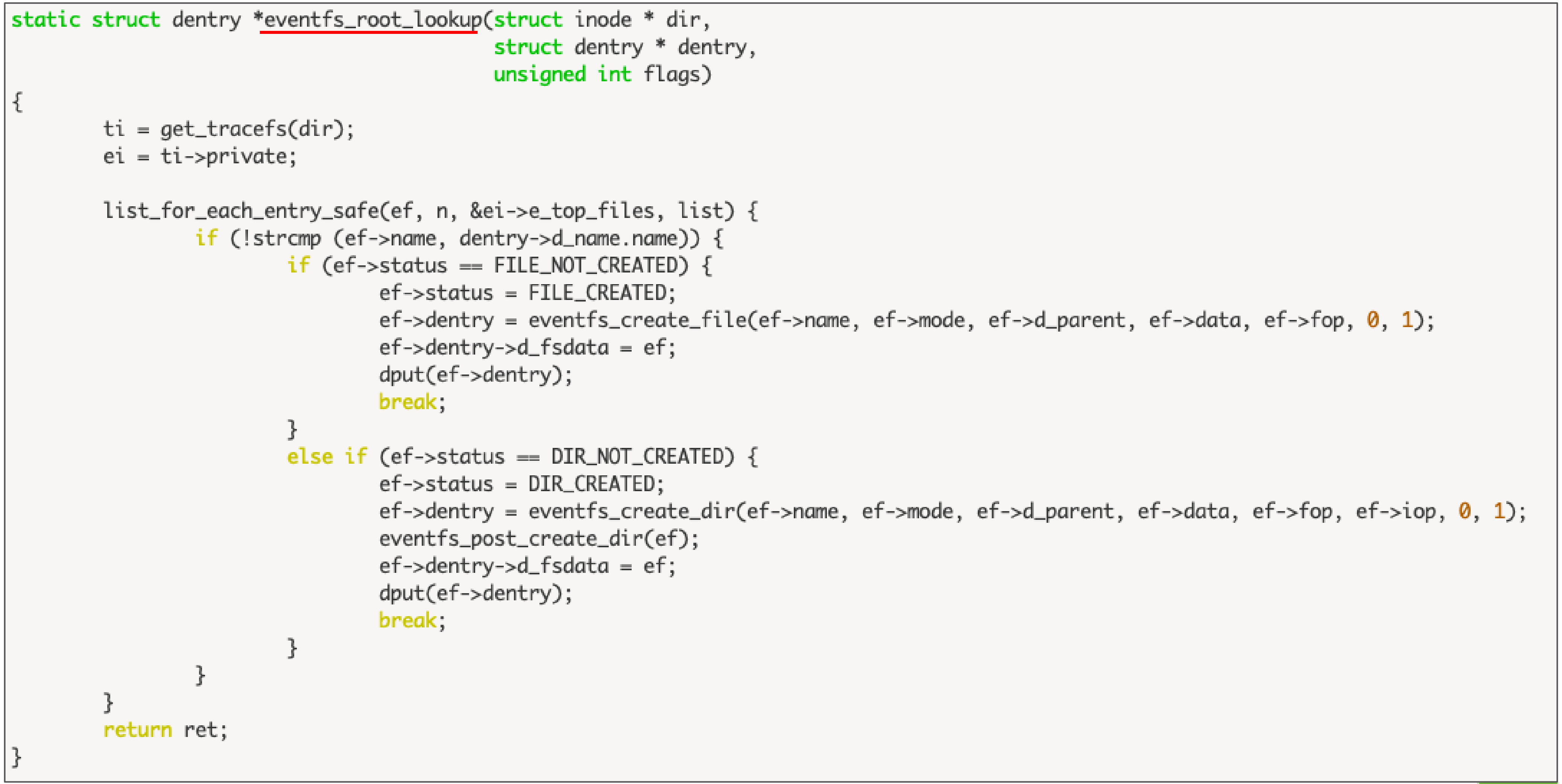
```
struct eventfs_file *eventfs_add_dir(const char *name, struct eventfs_file *ef_parent)
{
        struct eventfs_file *ef;

        ef = kzalloc(sizeof(struct eventfs_file), GFP_KERNEL);
        ef->ei = kzalloc(sizeof(struct eventfs_inode), GFP_KERNEL);

        ef->name = kstrdup(name, GFP_KERNEL);
        ef->mode = S_IFDIR | S_IRWXU | S_IRUGO | S_IXUGO;
        ef->iop = &eventfs_root_dir_inode_operations;
        ef->fop =  &eventfs_file_operations;
        ef->status = DIR_NOT_CREATED;
        ef->dentry = NULL;
        ef->d_parent = NULL;

        list_add_tail(&ef->list, &ef_parent->ei->e_top_files);
        return ef;
}
```

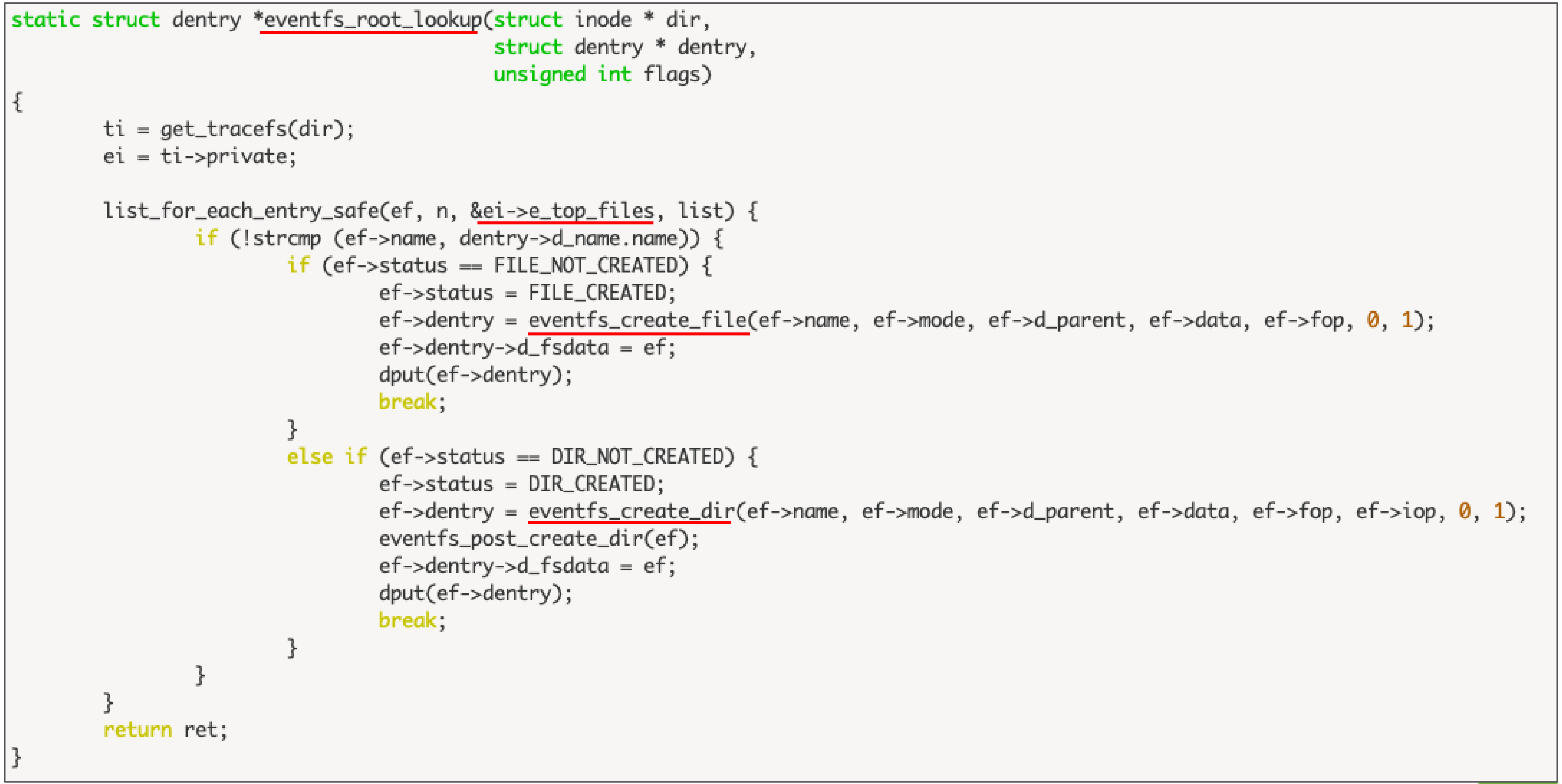
Eventfs: lookup and open



```
static struct dentry *eventfs_root_lookup(struct inode * dir,
                                          struct dentry * dentry,
                                          unsigned int flags)
{
        ti = get_tracefs(dir);
        ei = ti->private;

        list_for_each_entry_safe(ef, n, &ei->e_top_files, list) {
                if (!strcmp (ef->name, dentry->d_name.name)) {
                        if (ef->status == FILE_NOT_CREATED) {
                                ef->status = FILE_CREATED;
                                ef->dentry = eventfs_create_file(ef->name, ef->mode, ef->d_parent, ef->data, ef->fop, 0, 1);
                                ef->dentry->d_fsdata = ef;
                                dput(ef->dentry);
                                break;
                        }
                        else if (ef->status == DIR_NOT_CREATED) {
                                ef->status = DIR_CREATED;
                                ef->dentry = eventfs_create_dir(ef->name, ef->mode, ef->d_parent, ef->data, ef->fop, ef->iop, 0, 1);
                                eventfs_post_create_dir(ef);
                                ef->dentry->d_fsdata = ef;
                                dput(ef->dentry);
                                break;
                        }
                }
        }
        return ret;
}
```

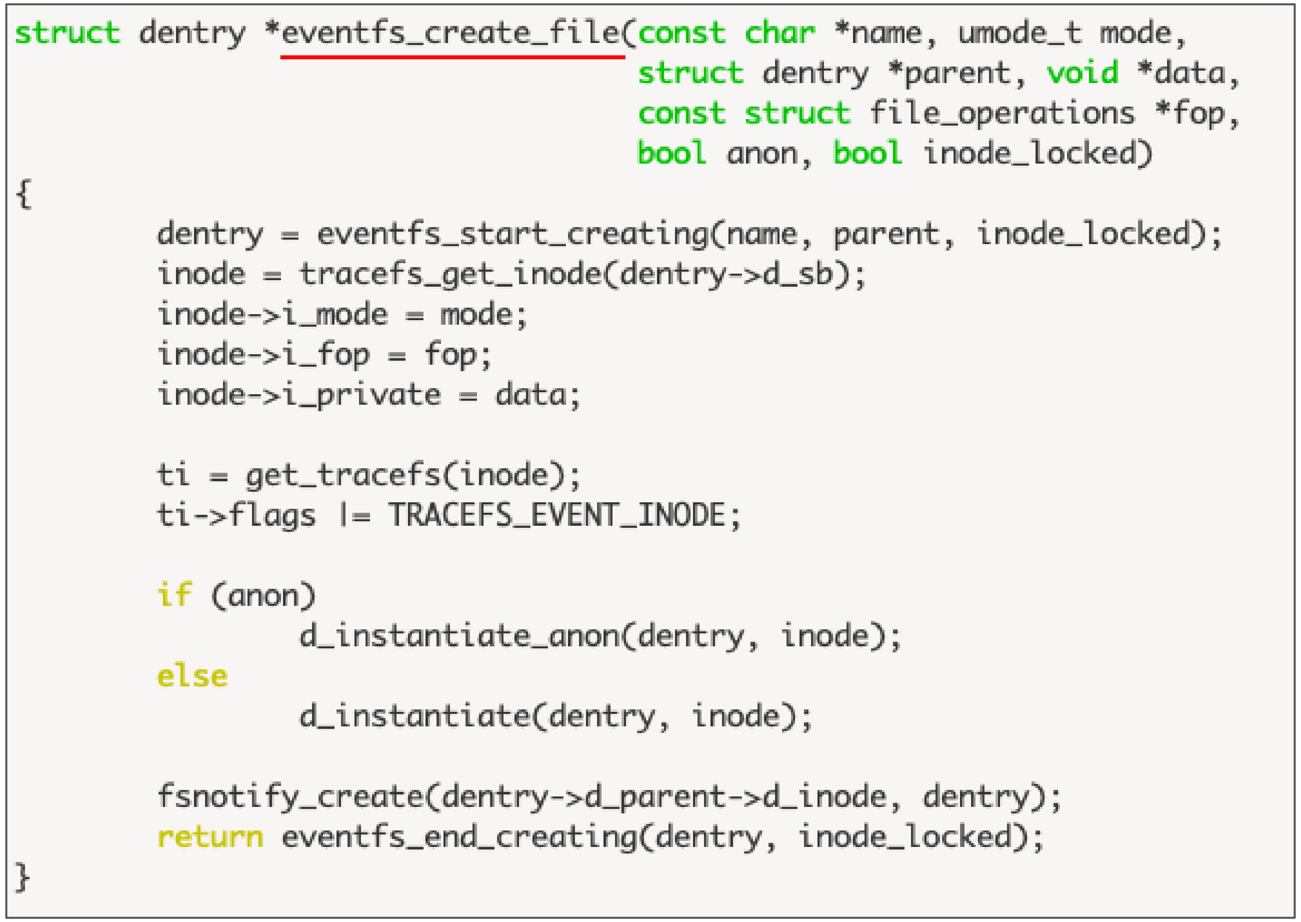
Eventfs: lookup and open



```
static struct dentry *eventfs_root_lookup(struct inode * dir,
                                          struct dentry * dentry,
                                          unsigned int flags)
{
        ti = get_tracefs(dir);
        ei = ti->private;

        list_for_each_entry_safe(ef, n, &ei->e_top_files, list) {
                if (!strcmp (ef->name, dentry->d_name.name)) {
                        if (ef->status == FILE_NOT_CREATED) {
                                ef->status = FILE_CREATED;
                                ef->dentry = eventfs_create_file(ef->name, ef->mode, ef->d_parent, ef->data, ef->fop, 0, 1);
                                ef->dentry->d_fsdata = ef;
                                dput(ef->dentry);
                                break;
                        }
                        else if (ef->status == DIR_NOT_CREATED) {
                                ef->status = DIR_CREATED;
                                ef->dentry = eventfs_create_dir(ef->name, ef->mode, ef->d_parent, ef->data, ef->fop, ef->iop, 0, 1);
                                eventfs_post_create_dir(ef);
                                ef->dentry->d_fsdata = ef;
                                dput(ef->dentry);
                                break;
                        }
                }
        }
        return ret;
}
```

Eventfs: create file or directory



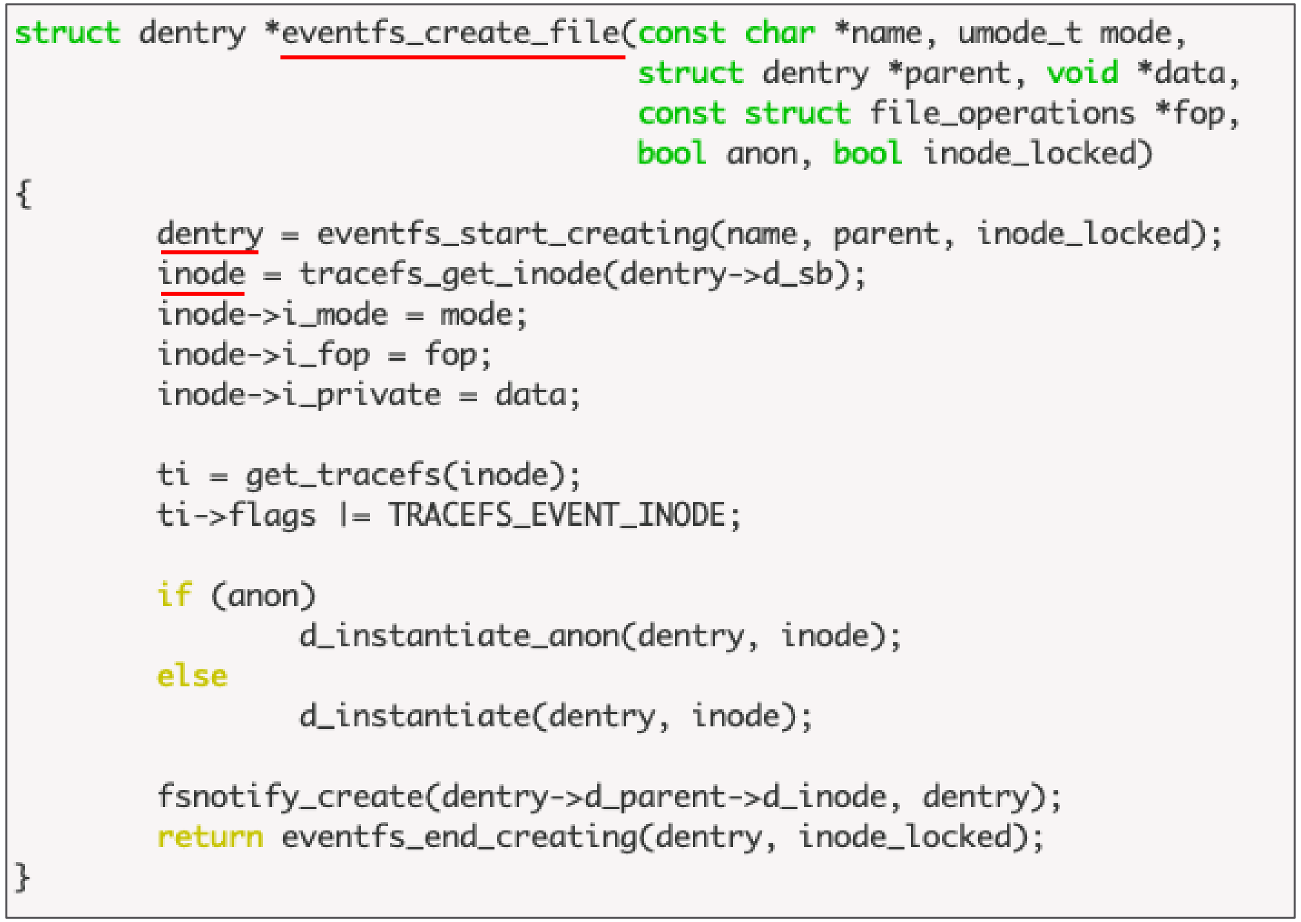
```
struct dentry *eventfs_create_file(const char *name, umode_t mode,
                                   struct dentry *parent, void *data,
                                   const struct file_operations *fop,
                                   bool anon, bool inode_locked)
{
        dentry = eventfs_start_creating(name, parent, inode_locked);
        inode = tracefs_get_inode(dentry->d_sb);
        inode->i_mode = mode;
        inode->i_fop = fop;
        inode->i_private = data;

        ti = get_tracefs(inode);
        ti->flags |= TRACEFS_EVENT_INODE;

        if (anon)
                d_instantiate_anon(dentry, inode);
        else
                d_instantiate(dentry, inode);

        fsnotify_create(dentry->d_parent->d_inode, dentry);
        return eventfs_end_creating(dentry, inode_locked);
}
```

Eventfs: create file or directory



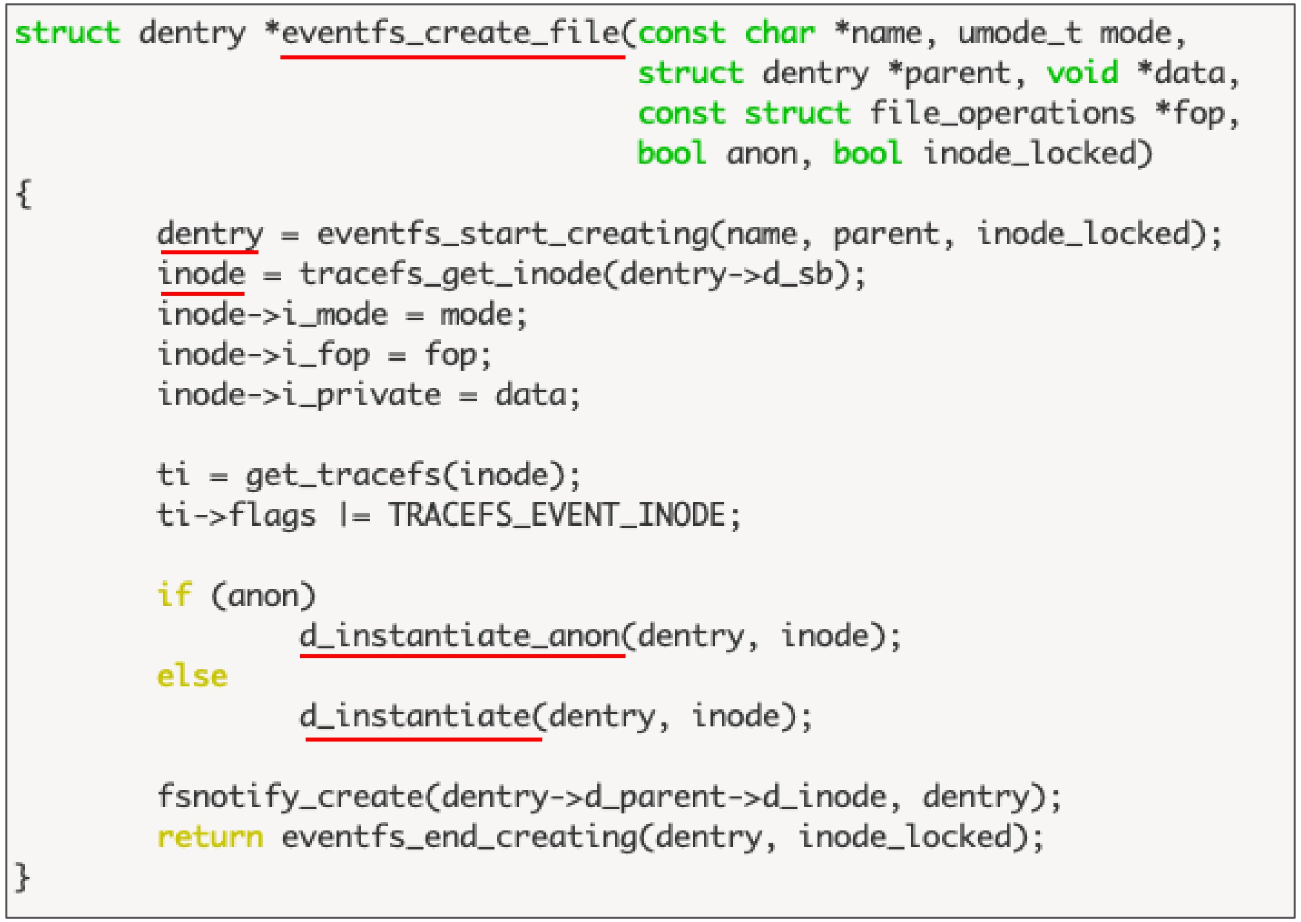
```
struct dentry *eventfs_create_file(const char *name, umode_t mode,
                                   struct dentry *parent, void *data,
                                   const struct file_operations *fop,
                                   bool anon, bool inode_locked)
{
        dentry = eventfs_start_creating(name, parent, inode_locked);
        inode = tracefs_get_inode(dentry->d_sb);
        inode->i_mode = mode;
        inode->i_fop = fop;
        inode->i_private = data;

        ti = get_tracefs(inode);
        ti->flags |= TRACEFS_EVENT_INODE;

        if (anon)
                d_instantiate_anon(dentry, inode);
        else
                d_instantiate(dentry, inode);

        fsnotify_create(dentry->d_parent->d_inode, dentry);
        return eventfs_end_creating(dentry, inode_locked);
}
```

Eventfs: create file or directory



```
struct dentry *eventfs_create_file(const char *name, umode_t mode,
                                   struct dentry *parent, void *data,
                                   const struct file_operations *fop,
                                   bool anon, bool inode_locked)
{
        dentry = eventfs_start_creating(name, parent, inode_locked);
        inode = tracefs_get_inode(dentry->d_sb);
        inode->i_mode = mode;
        inode->i_fop = fop;
        inode->i_private = data;

        ti = get_tracefs(inode);
        ti->flags |= TRACEFS_EVENT_INODE;

        if (anon)
                d_instantiate_anon(dentry, inode);
        else
                d_instantiate(dentry, inode);

        fsnotify_create(dentry->d_parent->d_inode, dentry);
        return eventfs_end_creating(dentry, inode_locked);
}
```

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

On-demand Eventfs:

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B

On-demand Eventfs:

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B

On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B
Files/Dirs in ‘events’	776 * 11742 = ~ 9MB

On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B
Files/Dirs in ‘events’	776 * 11742 = ~ 9MB

On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B
Files/Dirs in ‘events’	128 * 11742 = ~ 1.5MB

Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

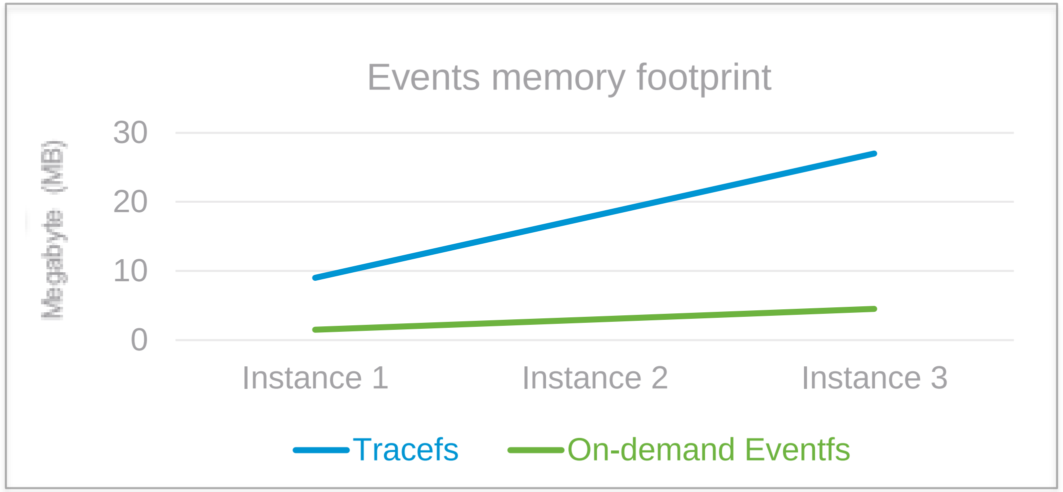
Theoretical values

inode + dentry	584 + 192 = 776B
Files/Dirs in ‘events’	776 * 11742 = ~ 9MB

On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B
Files/Dirs in ‘events’	128 * 11742 = ~ 1.5MB



Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B
Files/Dirs in ‘events’	776 * 11742 = ~ 9MB

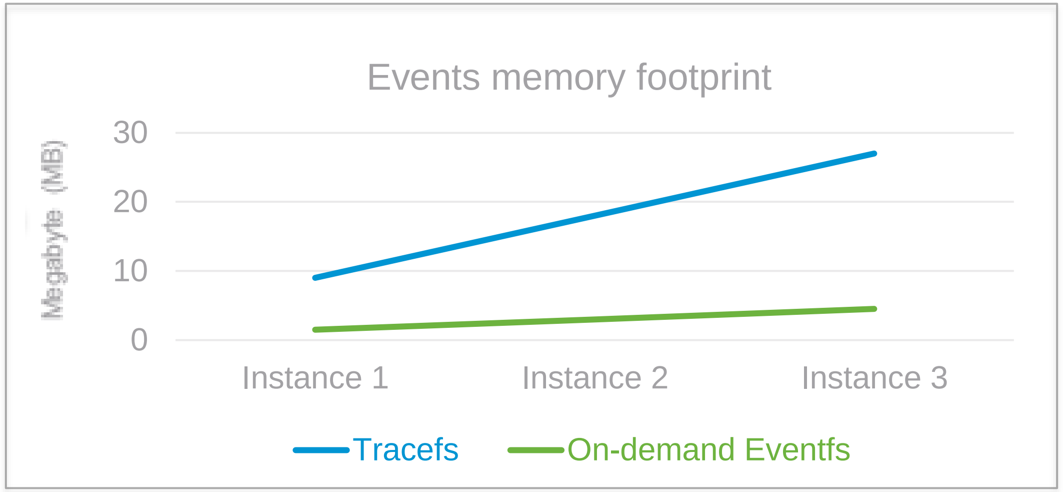
Practical values

Events Infrastructure	~ 9MB

On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B
Files/Dirs in ‘events’	128 * 11742 = ~ 1.5MB



Conclusion: Events Infrastructure memory footprint

Note: Following readings are from Linux Kernel v5.12, events directory is having 11742 files/directories and 8 CPUs

Tracefs:

Theoretical values

inode + dentry	584 + 192 = 776B
Files/Dirs in ‘events’	776 * 11742 = ~ 9MB

Practical values

Events Infrastructure	~ 9MB

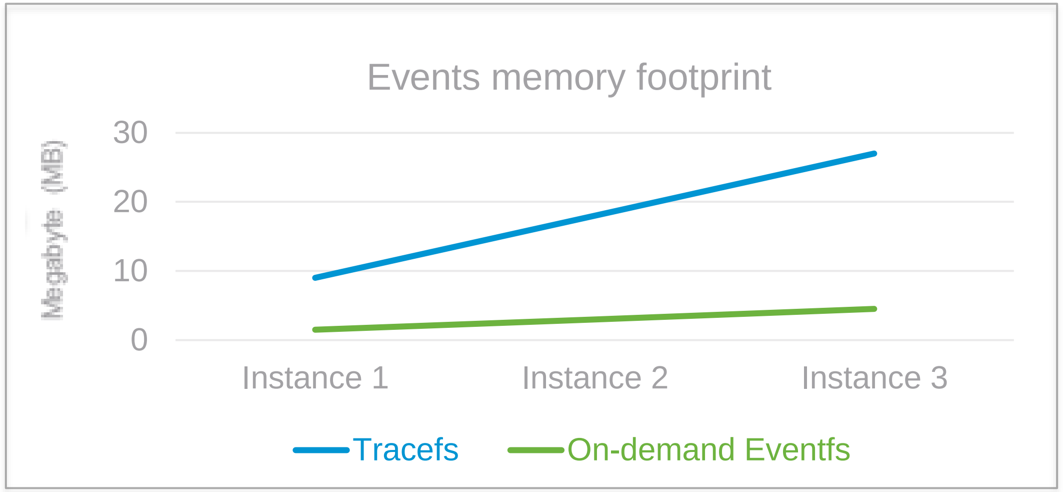
On-demand Eventfs:

Theoretical values

eventfs_inode + eventfs_file + name	80 + 16 + 32 = 128B
Files/Dirs in ‘events’	128 * 11742 = ~ 1.5MB

Practical values

‘Events Infrastructure’	~ 6MB



On going task:

- Analyzing why practical values are not matching with theoretical values.
- Enhance ‘On-Demand Eventfs’ to have one copy of Meta-data for Multiple Instances of Tracer.

Suggestions / Feedback:

- Is this the correct way to dynamically create files/directories?
- Any better approach to improve memory footprint of Linux Tracer.

Thanks