Programowanie Systemowe

Laboratorium 2 - Wprowadzenie

Magdalena Pastuła

Do wykonania ćwiczenia wykorzystano dostarczoną maszynę wirtualną z systemem Fedora w wersji 25.

1. LXR i dokumentacja.

1.1 Znajdź definicję struktury `file_operations`. Zapoznaj się z jej polami.

Znaleziona definicja struktury:

```
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t
*);
    ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
    ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
    int (*iopoll)(struct kiocb *kiocb, bool spin);
    int (*iterate) (struct file *, struct dir_context *);
    int (*iterate_shared) (struct file *, struct dir_context *);
    __poll_t (*poll) (struct file *, struct poll_table_struct *);
    long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
    long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    unsigned long mmap_supported_flags;
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *, fl_owner_t id);
    int (*release) (struct inode *, struct file *);
    int (*fsync) (struct file *, loff_t, loff_t, int datasync);
    int (*fasync) (int, struct file *, int);
    int (*lock) (struct file *, int, struct file_lock *);
    ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t
*, int);
    unsigned long (*get_unmapped_area)(struct file *, unsigned long,
unsigned long, unsigned long, unsigned long);
    int (*check_flags)(int);
    int (*flock) (struct file *, int, struct file_lock *);
    ssize_t (*splice_write)(struct pipe_inode_info *, struct file *, loff_t
*, size_t, unsigned int);
    ssize_t (*splice_read)(struct file *, loff_t *, struct pipe_inode_info
*, size_t, unsigned int);
    int (*setlease)(struct file *, long, struct file_lock **, void **);
    long (*fallocate)(struct file *file, int mode, loff_t offset,
                      loff_t len);
    void (*show_fdinfo)(struct seq_file *m, struct file *f);
#ifndef CONFIG_MMU
```

```
    unsigned (*mmap_capabilities)(struct file *);
#endif
    ssize_t (*copy_file_range)(struct file *, loff_t, struct file *,
                               loff_t, size_t, unsigned int);
    loff_t (*remap_file_range)(struct file *file_in, loff_t pos_in,
                               struct file *file_out, loff_t pos_out,
                               loff_t len, unsigned int remap_flags);
    int (*fadvise)(struct file *, loff_t, loff_t, int);
} __randomize_layout;
```

1.2 Spróbuj odnaleźć użycie tej struktury w systemie plików `hostfs`.

Znalezione użycie struktury `file_operations` w systemie plików `hostfs` w pliku `hostfs_kern.c`:

```
static const struct file_operations hostfs_file_fops = {
        .llseek         = generic_file_llseek,
        .splice_read    = generic_file_splice_read,
        .read_iter      = generic_file_read_iter,
        .write_iter     = generic_file_write_iter,
        .mmap           = generic_file_mmap,
        .open           = hostfs_open,
        .release        = hostfs_file_release,
        .fsync          = hostfs_fsync,
};

static const struct file_operations hostfs_dir_fops = {
        .llseek         = generic_file_llseek,
        .iterate_shared = hostfs_readdir,
        .read           = generic_read_dir,
        .open           = hostfs_open,
        .fsync          = hostfs_fsync,
};
```

1.3 Czy potrafisz zinterpretować zastosowanie poszczególnych pól?

Poszczególne pola odpowiadają operacjom możliwym do wykonania na pliku (`hostfs_file_fops`)i katalogu (`hostfs_dir_fops`). Możliwe operacje to między innymi:

- llseek -> ustawienie wskaźnika w pliku
- read_iter -> czytanie z pliku
- write_iter -> pisanie do pliku
- open -> otwarcie pliku
- release -> zamknięcie pliku

1.4 Zlokalizuj w dokumentacji (katalog `Documentation` lub wyszukiwarka) informacje o tym, jak przechodzić po strukturze katalogów w jądrze.

Głównym plikiem opisującym strukturę katalogów w jądrze jest plik `/filesystems/path_lookup.txt`.

1.5 Zlokalizuj plik `namei.h` a w nim funkcje `kern_path` i `user_path`. Czym się one różnią i kiedy mają zastosowanie?

Definicja funkcji `kern_path`:

```
int kern_path(const char *name, unsigned int flags, struct path *path)
{
    return filename_lookup(AT_FDCWD, getname_kernel(name),
                           flags, path, NULL);
}
```

Definicja funkcji `user_path`:

```
static inline int user_path_at(int dfd, const char __user *name,
                               unsigned flags, struct path *path)
{
    return user_path_at_empty(dfd, name, flags, path, NULL);
}
```

Funkcje `kern_path` oraz `user_path` wyszukują ścieżkę do pliku podanego w argumencie `name`. W tym celu funkcje te wyszukują strukturę `dentry`, która odpowiada danemu plikowi, a następnie zwracają tą strukturę poprzez argument `path`. Dodatkowo, zwiększany jest licznik odwołań do tych struktur.

Różnicą między tymi funkcjami jest środowisko: `kern_path` wyszukuje dla łańcucha znajdującego się w pamięci jądra, natomiast `user_path` dla łańcucha znajdujęcego się w pamięci użytkownika.

1.6 Znajdź definicję struktury `dentry`.

Znaleziona definicja struktury `dentry`:

```
struct dentry {
    /* RCU lookup touched fields */
    unsigned int d_flags;           /* protected by d_lock */
    seqcount_t d_seq;               /* per dentry seqlock */
    struct hlist_bl_node d_hash;    /* lookup hash list */
    struct dentry *d_parent;        /* parent directory */
    struct qstr d_name;
    struct inode *d_inode;          /* Where the name belongs to - NULL is
                                     * negative */
    unsigned char d_iname[DNAME_INLINE_LEN];        /* small names */
    /* Ref lookup also touches following */
    struct lockref d_lockref;       /* per-dentry lock and refcount */
    const struct dentry_operations *d_op;
    struct super_block *d_sb;       /* The root of the dentry tree */
    unsigned long d_time;           /* used by d_revalidate */
    void *d_fsdata;                 /* fs-specific data */

    union {
```

```
            struct list_head d_lru;         /* LRU list */
            wait_queue_head_t *d_wait;      /* in-lookup ones only */
    };
    struct list_head d_child;       /* child of parent list */
    struct list_head d_subdirs;     /* our children */
    /*
     * d_alias and d_rcu can share memory
     */
    union {
            struct hlist_node d_alias;      /* inode alias list */
            struct hlist_bl_node d_in_lookup_hash;  /* only for in-lookup
ones */
            struct rcu_head d_rcu;
    } d_u;
} __randomize_layout;
```

1.7 Co robi funkcja `dget` i po co? Znaleziona definicja funkcji `dget`:

```
/**
 *      dget, dget_dlock -      get a reference to a dentry
 *      @dentry: dentry to get a reference to
 *
 *      Given a dentry or %NULL pointer increment the reference count
 *      if appropriate and return the dentry. A dentry will not be
 *      destroyed when it has references.
 */
static inline struct dentry *dget_dlock(struct dentry *dentry)
{
    if (dentry)
        dentry->d_lockref.count++;
    return dentry;
}

static inline struct dentry *dget(struct dentry *dentry)
{
    if (dentry)
        lockref_get(&dentry->d_lockref);
    return dentry;
}
```

Jak wskazuje opis, funkcja ta zwiększa liczbę odwołań do struktury. Dzięki nowym odwołaniom struktura `dentry` nie zostanie usunięta.

2. Kompilacja jadra.

2.1 Zapoznaj się z plikiem `.config`.

Plik opisuje oraz zawiera konfigurację jądra.

2.2 Wykonaj polecenie `make help`.

Po wykonaniu polecenia `make help` wypisuje się przewodnik po Makefile'a do budowy jądra.

2.3 Co robi polecenie `make oldconfig`?

Polecenie `make oldconfig` dokonuje aktualizacji obecnej konfiguracji korzystając z pliku `.config.

2.4 Co robi polecenie `make menuconfig`?

Polecenie `make menuconfig` wyświetla menu graficzne, za pomocą którego można uaktualnić konfigurację w pliku `.config`

2.5 Ustaw dowolną, ale charakterystyczną wersję lokalną (`CONFIG_LOCALVERSION` albo `General setup/Local version`).

Ustawiono `JanTrzeci`.

2.6 Zrób małą zmianę w konfiguracji (np. włącz obsługę któregoś systemu plików).

Dodano obsługę systemu plików F2FS.

2.7 Co robi polecenie `make all`?

Polecenie `make all` buduje wszystkie targety oznaczone gwiazdką, czyli `vmlinux` i `modules` i skompresowane jądro (`bzImage`).

2.8 Zmierz czas kompilacji jądra po modyfikacji. Jakie informacje się wyświetlają podczas kompilacji?

Podczas kompilacji w terminalu wypisują się kolejno kompilowane pliki i moduły.

Zmierzony czas kompilacji: real 62m 38.067s user 194m 32.753s sys 30m 9.810s

2.9 Co robią polecenia `make modules_install` i `make install`?

Polecenie `make modules_install` instaluje wszystkie moduły jądra, natomiast komenda `make install` instaluje jądro w aktualnym systemie.

2.10 Zainstaluj jądro w systemie.

Wykonano komendy `make modules_install` oraz `make install`.

2.11 Zrestartuj system i uruchom nowe jądro.

Po restarcie i wpisaniu komendy `uname -a` wypisuje się następująca linia:

```
Linux ps2017 4.10.0JanTrzeci #1 SMP Fri Nov 13 09:04:03 CET 2020 x86_64
x86_64 x86_64 GNU/Linux
```

Jak widać, została zmieniona data ostatniej kompilacji oraz wypisuje się nazwa wpisana w `Local version`.

3. Moduł trivial_module.

Wynik po kompilacji modułu:

```
make -C /lib/modules/4.10.0JanTrzeci/build
M=/home/student/Downloads/trivial_module/intro/trivial_module modules
make[1]: Entering directory '/home/student/linux'
  CC [M]
/home/student/Downloads/trivial_module/intro/trivial_module/trivial.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC
/home/student/Downloads/trivial_module/intro/trivial_module/trivial.mod.o
  LD [M]
/home/student/Downloads/trivial_module/intro/trivial_module/trivial.ko
make[1]: Leaving directory '/home/student/linux'
```

3.1 Załaduj `insmod`.

Załadowano stosując instrukcję `insmod trivial.ko`

3.2 Wylistuj moduły.

Wypisane moduły:

```
Module                  Size  Used by
trivial                16384  0
xt_CHECKSUM            16384  1
ipt_MASQUERADE         16384  3
nf_nat_masquerade_ipv4    16384  1 ipt_MASQUERADE
nf_conntrack_netbios_ns    16384  1
nf_conntrack_broadcast    16384  1 nf_conntrack_netbios_ns
xt_CT                  16384  1
ip6t_rpfilter          16384  1
ip6t_REJECT            16384  2
nf_reject_ipv6         16384  1 ip6t_REJECT
xt_conntrack           16384  22
ip_set                 36864  0
nfnetlink              16384  1 ip_set
ebtable_nat            16384  1
ebtable_broute         16384  1
bridge                135168  1 ebtable_broute
stp                    16384  1 bridge
llc                    16384  2 bridge,stp
ip6table_nat           16384  1
nf_conntrack_ipv6      20480  12
nf_defrag_ipv6         36864  1 nf_conntrack_ipv6
nf_nat_ipv6            16384  1 ip6table_nat
ip6table_raw           16384  1
ip6table_mangle        16384  1
ip6table_security      16384  1
iptable_nat            16384  1
nf_conntrack_ipv4      16384  16
nf_defrag_ipv4         16384  1 nf_conntrack_ipv4
nf_nat_ipv4            16384  1 iptable_nat
```

```
nf_nat                 28672  3
nf_nat_ipv6,nf_nat_masquerade_ipv4,nf_nat_ipv4
nf_conntrack          131072  11
nf_conntrack_ipv6,nf_conntrack_ipv4,ipt_MASQUERADE,nf_conntrack_broadcast,n
f_conntrack_netbios_ns,xt_CT,nf_nat_ipv6,nf_nat_masquerade_ipv4,xt_conntrac
k,nf_nat_ipv4,nf_nat
libcrc32c              16384  1 nf_nat
iptable_raw            16384  1
iptable_mangle         16384  1
iptable_security       16384  1
ebtable_filter         16384  1
ebtables               36864  3 ebtable_filter,ebtable_nat,ebtable_broute
ip6table_filter        16384  1
ip6_tables             28672  5
ip6table_mangle,ip6table_filter,ip6table_security,ip6table_raw,ip6table_nat
vmw_vsock_vmci_transport    28672  2
vsock                  36864  3 vmw_vsock_vmci_transport
snd_seq_midi           16384  0
snd_seq_midi_event     16384  1 snd_seq_midi
kvm_intel             196608  0
kvm                   593920  1 kvm_intel
snd_ens1371            28672  5
snd_rawmidi            32768  2 snd_seq_midi,snd_ens1371
snd_ac97_codec        131072  1 snd_ens1371
ac97_bus               16384  1 snd_ac97_codec
snd_seq                65536  2 snd_seq_midi_event,snd_seq_midi
irqbypass              16384  1 kvm
crct10dif_pclmul       16384  0
snd_seq_device         16384  3 snd_seq,snd_rawmidi,snd_seq_midi
snd_pcm               118784  2 snd_ac97_codec,snd_ens1371
ppdev                  20480  0
crc32_pclmul           16384  0
snd_timer              32768  2 snd_seq,snd_pcm
vmw_balloon            20480  0
ghash_clmulni_intel    16384  0
joydev                 20480  0
snd                    81920  17
snd_seq,snd_ac97_codec,snd_timer,snd_rawmidi,snd_ens1371,snd_seq_device,snd
_pcm
pcspkr                 16384  0
soundcore              16384  1 snd
gameport               16384  1 snd_ens1371
nfit                   49152  0
parport_pc             28672  0
acpi_cpufreq           20480  0
tpm_tis                16384  0
tpm_tis_core           20480  1 tpm_tis
tpm                    40960  2 tpm_tis,tpm_tis_core
shpchp                 36864  0
nfsd                  335872  1
vmw_vmci               69632  2 vmw_balloon,vmw_vsock_vmci_transport
i2c_piix4              24576  0
auth_rpcgss            61440  1 nfsd
nfs_acl                16384  1 nfsd
```

```
lockd                  94208  1 nfsd
grace                  16384  2 nfsd,lockd
sunrpc                331776  7 auth_rpcgss,nfsd,nfs_acl,lockd
vmwgfx                241664  9
drm_kms_helper        155648  1 vmwgfx
ttm                    98304  1 vmwgfx
drm                   352256  12 vmwgfx,ttm,drm_kms_helper
e1000                 143360  0
mptspi                 24576  2
crc32c_intel           24576  1
scsi_transport_spi     32768  1 mptspi
mptscsih               40960  1 mptspi
ata_generic            16384  0
serio_raw              16384  0
pata_acpi              16384  0
mptbase               102400  2 mptscsih,mptspi
fjes                   73728  0
```

3.3 Obejrzyj komunikaty jądra: `dmesg`.

Ostatnie kilka linii komunikatów jądra:

```
[   16.589857] audit: type=1131 audit(1605264201.757:68): pid=1 uid=0
auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0
msg='unit=plymouth-read-write comm="systemd" exe="/usr/lib/systemd/systemd"
hostname=? addr=? terminal=? res=success'
[   17.953577] NET: Registered protocol family 40
[   22.487106] ip6_tables: (C) 2000-2006 Netfilter Core Team
[   22.816639] Ebtables v2.0 registered
[   23.855172] nf_conntrack version 0.5.0 (16384 buckets, 65536 max)
[   24.081306] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[   24.092062] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: None
[   25.477558] bridge: filtering via arp/ip/ip6tables is no longer
available by default. Update your scripts to load br_netfilter if you need
this.
[   25.766839] Netfilter messages via NETLINK v0.30.
[   25.871808] ip_set: protocol 6
[   27.892680] virbr0: port 1(virbr0-nic) entered blocking state
[   27.892683] virbr0: port 1(virbr0-nic) entered disabled state
[   27.892818] device virbr0-nic entered promiscuous mode
[   28.392613] virbr0: port 1(virbr0-nic) entered blocking state
[   28.392618] virbr0: port 1(virbr0-nic) entered listening state
[   28.501450] virbr0: port 1(virbr0-nic) entered disabled state
[   64.426793] perf: interrupt took too long (2505 > 2500), lowering
kernel.perf_event_max_sample_rate to 79000
[  120.126454] perf: interrupt took too long (3138 > 3131), lowering
kernel.perf_event_max_sample_rate to 63000
[  160.773030] perf: interrupt took too long (3951 > 3922), lowering
kernel.perf_event_max_sample_rate to 50000
[  216.153396] perf: interrupt took too long (5051 > 4938), lowering
kernel.perf_event_max_sample_rate to 39000
```

```
[  274.723108] perf: interrupt took too long (6483 > 6313), lowering
kernel.perf_event_max_sample_rate to 30000
[  630.758485] perf: interrupt took too long (8143 > 8103), lowering
kernel.perf_event_max_sample_rate to 24000
[  936.077694] perf: interrupt took too long (10195 > 10178), lowering
kernel.perf_event_max_sample_rate to 19000
[ 1106.225203] perf: interrupt took too long (12821 > 12743), lowering
kernel.perf_event_max_sample_rate to 15000
[ 1224.936587] trivial: loading out-of-tree module taints kernel.
[ 1224.936956] trivial: module verification failed: signature and/or
required key missing - tainting kernel
[ 1224.944378] Hello world! I'ma a trivial module!
```

3.4 Usuń: `rmmod`.

Wykonano komendę `rmmod trivial`.

3.5 Jeszcze raz obejrzyj komunikaty jądra.

Ostatnich kilka linii komunikatów jądra:

```
[   16.589857] audit: type=1131 audit(1605264201.757:68): pid=1 uid=0
auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0
msg='unit=plymouth-read-write comm="systemd" exe="/usr/lib/systemd/systemd"
hostname=? addr=? terminal=? res=success'
[   17.953577] NET: Registered protocol family 40
[   22.487106] ip6_tables: (C) 2000-2006 Netfilter Core Team
[   22.816639] Ebtables v2.0 registered
[   23.855172] nf_conntrack version 0.5.0 (16384 buckets, 65536 max)
[   24.081306] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[   24.092062] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow
Control: None
[   25.477558] bridge: filtering via arp/ip/ip6tables is no longer
available by default. Update your scripts to load br_netfilter if you need
this.
[   25.766839] Netfilter messages via NETLINK v0.30.
[   25.871808] ip_set: protocol 6
[   27.892680] virbr0: port 1(virbr0-nic) entered blocking state
[   27.892683] virbr0: port 1(virbr0-nic) entered disabled state
[   27.892818] device virbr0-nic entered promiscuous mode
[   28.392613] virbr0: port 1(virbr0-nic) entered blocking state
[   28.392618] virbr0: port 1(virbr0-nic) entered listening state
[   28.501450] virbr0: port 1(virbr0-nic) entered disabled state
[   64.426793] perf: interrupt took too long (2505 > 2500), lowering
kernel.perf_event_max_sample_rate to 79000
[  120.126454] perf: interrupt took too long (3138 > 3131), lowering
kernel.perf_event_max_sample_rate to 63000
[  160.773030] perf: interrupt took too long (3951 > 3922), lowering
kernel.perf_event_max_sample_rate to 50000
[  216.153396] perf: interrupt took too long (5051 > 4938), lowering
kernel.perf_event_max_sample_rate to 39000
[  274.723108] perf: interrupt took too long (6483 > 6313), lowering
```

```
kernel.perf_event_max_sample_rate to 30000
[  630.758485] perf: interrupt took too long (8143 > 8103), lowering
kernel.perf_event_max_sample_rate to 24000
[  936.077694] perf: interrupt took too long (10195 > 10178), lowering
kernel.perf_event_max_sample_rate to 19000
[ 1106.225203] perf: interrupt took too long (12821 > 12743), lowering
kernel.perf_event_max_sample_rate to 15000
[ 1224.936587] trivial: loading out-of-tree module taints kernel.
[ 1224.936956] trivial: module verification failed: signature and/or
required key missing - tainting kernel
[ 1224.944378] Hello world! I'ma a trivial module!
[ 1432.358171] perf: interrupt took too long (16117 > 16026), lowering
kernel.perf_event_max_sample_rate to 12000
[ 1593.403846] Oh no, why are you doing this to me? Argh!
```

Komenda `dmseg` to komenda służaca do wyświetlenia zawartości lub zmiany bufora kołowego jądra.

4. Uruchomianie jądra w QEMU.

4.1 Skompiluj jądro, podobnie jak we wcześniejszym zadaniu.

Przed kompilacją zmieniono nazwę wersji lokalnej na `QEMU`.

4.2 Uruchom:

```
qemu-system-x86_64 \
   -kernel arch/x86/boot/bzImage \
   -hda ~/fs/CentOS6.x-AMD64-root_fs \
   -append 'root=/dev/sda'
```

4.3 Zaloguj się do systemu. Login: root, hasło: puste.

Po zalogowaniu się i wykonaniu komendy `uname -a` ukazuje się następująca informacja:

```
Linux localhost.localdomian 4.10.0QEMU #2 SMP Fri Nov 13 12:21:26 CET 2020
x86_64 x86_64 GNU/Linux
```

Zatem widać, że w QEMU została uruchomiona skompilowana wersja jądra.

4.4 Zakończ pracę z gościem.

Aby zakończyć pracę wykonano komendę `poweroff`. Komenda `exit` powoduje tylko wylogowanie się.

5. Kompilacja jądra UML.

5.1 Wykonaj komendę `make ARCH=um defconfig`. Co ona robi?

Informacje wypisujące się po wykonaniu komendy:

```
*** Default configuration is based on 'x86_64_defconfig'
kernel/time/Kconfig:155:warning: range is invalid
#
# configuration written to .config
#
```

Komenda ta zmienia konfigurację jądra na domyślną dla architektury wyszczególnionej w argumencie ARCH. W tym przypadku architekturą tą jest tryb użytkownika.

5.2 Skompiluj jądro komendą `make ARCH=um`. Ile czasu zajęła kopilacja?

Czas kompilacji: real 1m28.880s user 4m25.051s sys 0m42.474s

5.3 Uruchom `./vmlinux ubd0=~/fs/CentOS6.x-AMD64-root_fs`.

Po uruchomieniu pojawia sie informacja:

```
CentOS release 6.6 (Final)
Kernel 4.10.0 on an x86_64
```

5.4 Zaloguj się do systemu. Login i hasło jak poprzednio.

Po zalogowaniu się i wpisaniu komendy `uname -a` pokazuje się następująca informacja:

```
Linux localhost.localdomain 4.10.0 #3 Fri Nov 13 12:58:10 CET 2020 x86_64
x86_64 x86_64 GNU/Linux
```

Ponownie, widać, że środowisko zostało uruchomione ze skompilowanego jądra.

5.5 Zamontuj hostfs za pomocą komendy `mount none /host -t hostfs` Co pojawiło się po zamontowaniu systemu plików?

Po wpisaniu komendy nic się nie wypisuje w terminalu, natomiast w folderze `/host` pojawia się następująca zawartość:

```
1    boot  etc   lib    lost+found  mnt  proc  run   srv  tmp  var
bin  dev   home  lib64  media       opt  root  sbin  sys  usr
```

Po każdej komendzie dodatkowo wypisuje się informacja o błędzie:

```
modprobe: FATAL: Could not load /lib/modules/4.10.0/modules.dep: No such
file or directory
```

5.6 Jakie procesy są widoczne w gościu i po stronie hosta?

Procesy widoczne w gościu:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  8.2  19284  2256 ?        Ss   07:03   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    07:03   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    07:03   0:00
[kworker/0:0]
root         4  0.0  0.0      0     0 ?        S<   07:03   0:00
[kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    07:03   0:00
[ksoftirqd/0]
root         7  0.0  0.0      0     0 ?        S<   07:03   0:00 [lru-add-
drain]
root         8  0.0  0.0      0     0 ?        S    07:03   0:00
[kdevtmpfs]
root         9  0.0  0.0      0     0 ?        S<   07:03   0:00 [netns]
root        10  0.0  0.0      0     0 ?        S    07:03   0:00
[oom_reaper]
root        11  0.0  0.0      0     0 ?        S    07:03   0:00
[kworker/u2:1]
root        45  0.0  0.0      0     0 ?        S<   07:03   0:00
[writeback]
root        46  0.0  0.0      0     0 ?        S<   07:03   0:00 [crypto]
root        47  0.0  0.0      0     0 ?        S    07:03   0:00
[kworker/0:1]
root        49  0.0  0.0      0     0 ?        S<   07:03   0:00 [bioset]
root        51  0.0  0.0      0     0 ?        S<   07:03   0:00 [kblockd]
root        73  0.0  0.0      0     0 ?        S    07:03   0:00 [kswapd0]
root        74  0.0  0.0      0     0 ?        S<   07:03   0:00 [bioset]
root       227  0.0  0.0      0     0 ?        S<   07:03   0:00 [bioset]
root       236  0.0  0.0      0     0 ?        S    07:03   0:00
[jbd2/ubda-8]
root       237  0.0  0.0      0     0 ?        S<   07:03   0:00 [ext4-rsv-
conver]
root       305  0.0  5.4  10700  1488 ?        S<s  07:03   0:00
/sbin/udevd -d
root       400  0.0  5.2  10696  1432 ?        S<   07:03   0:00
/sbin/udevd -d
root       406  0.0  0.0      0     0 ?        S<   07:03   0:00
[kworker/0:1H]
root       576  0.0  9.1  66268  2500 ?        Ss   07:03   0:00
/usr/sbin/sshd
root       589  0.0  9.1  52352  2492 ?        Ss   07:03   0:00 login --
root
root       595  0.0  9.6  11488  2632 tty0     Ss   07:03   0:00 -bash
root       606  0.0  0.0      0     0 ?        S    07:06   0:00
[kworker/u2:2]
root       637  0.0  6.8  13372  1860 tty0     R+   07:16   0:00 ps aux
```

Procesy widoczne w hoście po odfiltrowaniu rekordów ze słowem `vmlinux`:

```
student  60942  1.0  1.0  45552 33640 pts/0    S+   13:03   0:09 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  60947  0.0  1.0  45552 33640 pts/0    S+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  60948  0.0  1.0  45552 33640 pts/0    S+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  60949  0.0  1.0  45552 33640 pts/0    S+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  60950  0.0  0.0  14908  2264 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  61080  0.0  0.0  14140   696 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  61224  0.0  0.0  14084   788 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  61523  0.0  0.0  13860   740 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  61549  0.0  0.0  15144  1320 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  61556  0.0  0.0  15284  1892 pts/0    t+   13:03   0:00 ./vmlinux
ubd0=/home/student/fs/CentOS6.x-AMD64-root_fs
student  62038  0.0  0.0 119372   912 pts/1    S+   13:19   0:00 grep --
color=auto vmlinu
```

Zatem część procesów z gościa jest przenoszona zapewne na hosta.

5.7 Zakończ pracę z gościem.

Analogicznie, jak w przypadku QEMU, aby wyjść ze środowiska należało wywołać komendę `poweroff`, ponieważ `exit` powoduje tylko wylogowanie się.