

SO2 Cheat Sheet

Compilarea nucleului

make menuconfig – configurarea nucleului Linux
make – compilarea nucleului Linux
make modules_install – instalează modulele
make install – instalează imagina de kernel
build – compilarea nucleului Windows
/boot/grub/menu.lst – fișierul de configurare a GRUB
C:\boot.ini – configurare booting pe Windows

Aplicații utile în kernel programming

GCC, GNU Make, GDB – suita standard pe Linux
Windows Driver Kit (WDK) – suita de driver development pe Windows
Windows Research Kernel (WRK) – o parte din sursele nucleului Windows
WinDbg, LiveKd – utilitare de debugging în Windows
Linux Cross Reference (LXR) – interfață web pentru source browsing

Crearea unui modul simplu în Linux

kernel.h, init.h, module.h – headere necesare
MODULE_[DESCRIPTION|AUTHOR|LICENSE] – informații modul
static int init_func(void) – entry point
static void exit_func(void) – exit point
module_init, module_exit – specificare funcții entry / exit point

Crearea unui modul simplu în Windows

ntddk.h – header necesar
NTSTATUS DriverEntry(PDRIVER_OBJECT driver, PUNICODE_STRING registry) – entry point
driver->DriverUnload – înregistrare exit point

Toolchains

make, kbuild – compilare module Linux
lsmod, insmod, rmmod – lucrul cu module în Linux
printk, addr2line, objdump, netconsole – debugging Linux
nmake, sources – compilare module Windows
driver [list, load, unload] – lucrul cu module în Windows
WinDbg, !analyze -v – debugging Windows

Alocare memorie

Linux
void *kmalloc(size_t size, int flags); – alocare memorie
void *kzalloc(size_t size, int flags); – alocare memorie inițializată la zero
void kfree(const void *mem); – eliberare memorie

Windows

```
PVOID ExAllocatePoolWithTag(
    IN POOL_TYPE PoolType,
    IN SIZE_T NumberOfBytes,
    IN ULONG Tag); – alocare memorie
VOID ExFreePoolWithTag(
    IN PVOID P,
    IN ULONG Tag); – eliberare memorie
```

Liste

Linux

```
struct list_head – listă
list_add(struct list_head *new, struct list_head *head)
    – inserează new după head
list_del(struct list_head *entry)
    – șterge un element din listă
list_empty(struct list_head *head)
    – verifică dacă o listă este goală
list_entry(ptr, type, member)
    – întoarce structura ce conține list_head-ul
list_for_each(pos, head)
    – parcurge o listă
list_for_each_safe(pos, head)
    – parcurge o listă (safe pentru ștergerea unui element)
```

Windows

```
SINGLE_LIST_ENTRY – listă
LIST_ENTRY – listă dublu înlănțuită
VOID PushEntryList(
    IN PSINGLE_LIST_ENTRY ListHead
    IN PSINGLE_LIST_ENTRY Entry);
    – inserează Entry
VOID PopEntryList(
    IN PSINGLE_LIST_ENTRY ListHead);
    – șterge primul element din listă
```

Locking

Linux

```
spinlock_t – tip spinlock
spin_lock_init(spinlock_t *lock); – inițializare spinlock
spin_lock(spinlock_t *lock); – obținere lock
spin_unlock(spinlock_t *lock); – eliberare lock
struct semaphore – tip semafor
void sema_init(struct semaphore *sem, int val);
    – inițializare semafor la valoarea val
void down(struct semaphore *sem); – decrementare semafor
void up(struct semaphore *sem); – incrementare semafor
atomic_t – tip atomic
atomic_set(atomic_t *v, int i)
int atomic_read(atomic_t *v);
void atomic_add(int i, atomic_t *v);
void atomic_sub(int i, atomic_t *v);
void atomic_inc(atomic_t *v);
```

```
void atomic_dec(atomic_t *v);
int atomic_inc_and_test(atomic_t *v);
int atomic_dec_and_test(atomic_t *v);
Windows
KSPIN_LOCK – tip spinlock
VOID KeInitializeSpinLock(
    IN PKSPIN_LOCK SpinLock); – inițializare spinlock
VOID KeAcquireSpinLock(
    IN PKSPIN_LOCK SpinLock,
    OUT PKIRQL OldIrql); – obținere lock
VOID KeReleaseSpinLock(
    IN PKSPIN_LOCK SpinLock,
    IN KIRQL NewIrql); – eliberare lock
KESEMAPHORE – tip semafor
VOID KeInitializeSemaphore(
    IN PRKSEMAPHORE Semaphore,
    IN LONG Count,
    IN LONG Limit); – inițializare semafor
NTSTATUS KeWaitForSingleObject(...) – decrementare semafor
LONG KeReleaseSemaphore(...) – incrementare semafor
InterlockedCompareExchange(...)
InterlockedDecrement(...)
InterlockedExchange(...)
InterlockedExchangeAdd(...)
InterlockedIncrement(...)
```

Device drivers

Linux

```
mknod <fișier> <c/b> <major> <minor>
int register_chrdev_region(dev_t first, unsigned int count,
char *name)
void unregister_chrdev_region(dev_t first, unsigned int
count)
void cdev_init(struct cdev *cdev, struct file_operations
*fops)
int cdev_add(struct cdev *dev, dev_t num, unsigned int
count)
void cdev_del(struct cdev *dev)
put_user(type val, type *address)
get_user(type val, type *address)
unsigned long copy_to_user(void __user *to, const void
*from, unsigned long n)
unsigned long copy_from_user(void *to, const void __user
*from, unsigned long n)
int (*open) (struct inode *, struct file *)
ssize_t (*read) (struct file *, char __user *, size_t,
loff_t *);
ssize_t (*write) (struct file *, const char __user *,
size_t, loff_t *);
int (*ioctl) (struct inode *, struct file *, unsigned int,
unsigned long);
int (*release) (struct inode *, struct file *);
```

Waitqueues

Linux

```
DECLARE_WAIT_QUEUE_HEAD(wait_queue_head_t *q);
void init_waitqueue_head(wait_queue_head_t *q);
int wait_event(wait_queue_head_t *q, int condition);
int wait_event_interruptible(wait_queue_head_t *q, int
condition);
int wait_event_timeout(wait_queue_head_t *q, int condition,
int timeout);
int wait_event_interruptible_timeout(wait_queue_head_t *q,
int condition, int timeout);
void wake_up(wait_queue_head_t *q);
void wake_up_interruptible(wait_queue_head_t *q);
```