Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Λειτουργικά Συστήματα (ECE318)

Ακαδημαϊκό Έτος 2020-2021

Εισαγωγικός Οδηγός στα Linux Kernel Modules

Τελευταία Ενημέρωση: 16 Μαρτίου 2021

Περιεχόμενα

1 Linux Kernel Modules

Τα kernel modules στο λειτουργικό σύστημα Linux είναι τμήματα κώδικα που μας επιτρέπουν να προσθέσουμε επιπλέον λειτουργικότητα στον πυρήνα του λειτουργικού συστήματος, χωρίς να απαιτείται η εκ νέου μεταγλώττιση του πυρήνα και η επανεκκίνηση του λειτουργικού συστήματος. Ο παρών οδηγός έχει ως στόχο να σας παρέχει οδηγίες για τις διαδικασίες ανάπτυξης, μεταγλώττισης και φόρτωσης ενός module στον πυρήνα του λειτουργικού συστήματος.

1.1 Υλικό για Μελέτη

Κεφάλαιο 2 του βιβλίου ”Linux Devices Drivers”, 3η έκδοση.

2 Διαδικασία Ανάπτυξης ενός Kernel Module

Η διαδικασία ανάπτυξης ενός kernel module διαφέρει σημαντικά από την διαδικασία που ακολουθούμε για την ανάπτυξη μίας εφαρμογής επιπέδου χρήστη. Παρακάτω παρουσιάζεται ο πηγαίος κώδικας ενός πολύ απλού kernel module, το οποίο εκτυπώνει δύο μηνύματα.

Παράδειγμα ενός απλού kernel module

```
#include <linux/init.h>
#include <linux/module.h>

MODULE_LICENSE("Dual BSD/GPL");

static int hello_init(void) {
    printk(KERN_ALERT "Hello, world\n");
    return 0;
}

static void hello_exit(void) {
    printk(KERN_ALERT "Goodbye, cruel world\n");
}

module_init(hello_init);
module_exit(hello_exit);
```

Προκειμένου να μεταγλωττιστεί επιτυχώς, ο κώδικας ενός kernel module θα πρέπει κατ’ ελάχιστον:

- Να συμπεριλαμβάνει το αρχείο κεφαλίδας *linux/module.h*.
- Να ορίζει τις συναρτήσεις που θα καλούνται κατά την φόρτωση και την αφαίρεση του module από τον πυρήνα. Αυτό γίνεται με χρήση των macros *module_init()* και *module_exit()*.
- Να ορίζει την άδεια βάσει της οποίας διανέμεται το module. Αυτό πραγματοποιείται με χρήση του macro *MODULE_LICENSE()*.

Ο κώδικας που παρουσιάστηκε παραπάνω ορίζει τις συναρτήσεις *hello_init()* και *hello_exit()*, οι οποίες καλούνται κατά την φόρτωση και την αφαίρεση του module από τον πυρήνα αντίστοιχα. Επίσης, παρατηρήστε την χρήση του *MODULE_LICENSE()* για τον ορισμό της άδειας του module. Αυτό το macro εισάγεται για ”νομικούς” λόγους. Σε περίπτωση που δεν το χρησιμοποιήσουμε, το σύστημα εμφανίζει warnings κατά την φόρτωση του module. Τέλος, παρατηρήστε τη χρήση της συνάρτησης *printk()*. Η συνάρτηση αυτή ορίζεται στον πυρήνα του λειτουργικού και είναι διαθέσιμη προς χρήση από τα διάφορα modules. Προσφέρει παρόμοια λειτουργικότητα με την συνάρτηση *printf()* αλλά είναι απαραίτητη καθώς ο πυρήνας δεν μπορεί να χρησιμοποιήσει συναρτήσεις της C library (όπως η *printf()*, η *malloc()* κλπ). Τα μηνύματα που εκτυπώνονται με χρήση της *printk()* αποθηκεύονται στα αρχεία καταγραφής μηνυμάτων (logs) του πυρήνα. Για να δείτε αυτά τα μηνύματα, εκτελέστε την εντολή *dmesg* σε ένα terminal. Επειδή στα αρχεία καταγραφής υπάρχει μεγάλο πλήθος μηνυμάτων, για να δείτε τα τελευταία μηνύματα ανακατευθύνετε το αποτέλεσμα της *dmesg* στην εντολή *tail* (*dmesg* | *tail*).

3 Μεταγλώττιση και Φόρτωση ενός Module

Έστω ότι έχουμε αποθηκεύσει τον κώδικα του module που παρουσιάστηκε παραπάνω στο αρχείο με όνομα *hello_module.c*. Για την μεταγλώττιση του κώδικα θα χρησιμοποιήσουμε το σύστημα kbuild που παρέχεται με τον πυρήνα του Linux ώστε να παράγουμε το module με όνομα *hello_module.ko*. Για την μεταγλώττιση, θα χρησιμοποιήσουμε το παρακάτω Makefile.

Παράδειγμα Makefile

```
obj-m := hello_module.o

KERNELDIR ?= /lib/modules/$(shell uname -r)/build
PWD := $(shell pwd)

default:
    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules

clean:
        rm -f *.o *.ko *.mod.c
```

Στην πρώτη γραμμή ορίζουμε ότι το module θα παραχθεί από το object file hello_module.o (το οποίο θα προκύψει από τη μεταγλώττιση του αρχείου hello_module.c). Σε περίπτωση που το module δημιουργείται από περισσότερα του ενός αρχεία, έστω file_1.c και file_2.c, η πρώτη γραμμή θα τροποποιούνταν ως εξής:

obj-m := module.o
module-objs := file_1.o file_2.o

Το παραπάνω σημαίνει ότι θα φτιαχτεί ένα object file με όνομα module.o, το οποίο θα δημιουργηθεί με σύνθεση των object files file_1.o και file_2.o (τα τελευταία, με τη σειρά τους, θα δημιουργηθούν με τη μεταγλώττιση των αντίστοιχων αρχείων .c).

Αφού γράψουμε make, και με την προϋπόθεση ότι όλα πήγαν καλά, θα φτιαχτεί το αρχείο του module, το οποίο θα έχει κατάληξη .ko.

Το επόμενο βήμα είναι η εισαγωγή του module στον πυρήνα του λειτουργικού συστήματος. Αυτό μπορεί να πραγματοποιηθεί χρησιμοποιώντας την εντολή *insmod*. Ανοίξτε ένα terminal, μεταβείτε στον φάκελο που έχετε αποθηκεύσει τα αρχεία του module και εκτελέστε την εντολή *sudo insmod hello_module.ko*, για να εισάγετε το module hello_module στον πυρήνα. Σε περίπτωση που η εισαγωγή είναι επιτυχημένη, θα μπορείτε να δείτε το μήνυμα ”Hello, world” ανάμεσα στα μηνύματα που εμφανίζονται εκτελώντας την εντολή *dmesg | tail* σε ένα terminal.

Οποιαδήποτε στιγμή το θελήσετε μπορείτε να δείτε ποια modules είναι φορτωμένα εκτελώντας την εντολή *lsmod*. Αν θέλετε να ψάξετε για ένα συγκεκριμένο module (π.χ. το hello_module), μπορείτε να ανακατευθύνετε την έξοδο της εντολής *lsmod* στην εντολή *grep* (π.χ. *lsmod | grep hello_module*).

Τέλος, για την αφαίρεση του module χρησιμοποιείται η εντολή *rmmod*. Σε αντίθεση με την εντολή *insmod*, προκειμένου να εκτελέσετε την εντολή rmmod δε χρειάζεται να βρίσκεστε στο directory όπου βρίσκονται τα αρχεία του module. Εκτελέστε την εντολή *sudo rmmod hello_module.ko* σε ένα terminal. Σε περίπτωση που δεν εμφανιστεί κάποιο μήνυμα, η αφαίρεση έχει πραγματοποιηθεί με επιτυχία και μπορείτε να δείτε το μήνυμα ”Goodbye, cruel world” ανάμεσα στα μηνύματα που εμφανίζονται εκτελώντας την εντολή *dmesg | tail* σε ένα terminal.