Linux Kernel: Adding your own sysctls

Vitalik Nikolyenko - vnik5287@uni.sydney.edu.au

Linux Kernel: Adding your own sysctls

Sysctl entries are represented by an array of `struct proc_dir_entry`, which is defined in `<linux/sysctl.h>`:

```
struct ctl_table
{
        int ctl_name;                   /* Binary ID */
        const char *procname;           /* Text ID for /proc/sys, or zero */
        void *data;
        int maxlen;
        mode_t mode;
        struct ctl_table *child;
        struct ctl_table *parent;       /* Automatically set */
        proc_handler *proc_handler;     /* Callback for text formatting */
        ctl_handler *strategy;          /* Callback function for all r/w */
        void *extra1;
        void *extra2;
};
```

ctl_name is a unique (within the current sysctl level) numeric value used by sysctl utility.

procname specifies the name of the sysctl file under /proc/sys.

data is a pointer to data for use by proc_handler.

maxlen is the maximum size in bytes of the data.

mode specifies the file permissions for the /proc/sys file.

child is a pointer to the child sysctl table if this entry is a directory or NULL.

proc_handler - the text handler routine.

extra1 and extra2 are extra pointers usable by some proc handler routines.

Each array entry can either be a parent or a child. Parent structs represent directories within `/proc/sys/` and should set their `.child` member to point to the child table. Child entries are terminal and represent files; their .child member is set to NULL. Note that each array must be terminated by a NULL entry.

The following example will create /proc/sys/kernel/sample file that accepts a value in range $[0 : 5]$ and assigns it to `global_var`:

```
static int global_var;
static int min_val = 0;
static int max_val = 5;

static struct ctl_table sample_child_table[] = {
        {
                .ctl_name       = CTL_UNNUMBERED,
                .procname       = "sample",
                .maxlen         = sizeof(int),
                .mode           = 0644,
                .data           = &global_var,
                .proc_handler   = &proc_dointvec_minmax,
                .extra1         = &min_val,
                .extra2         = &max_val,
        },
        {}
};

static struct ctl_table sample_parent_table[] = {
{
                .ctl_name       = CTL_KERN,
                .procname       = "kernel",
                .mode           = 0555,
                .child          = sample_child_table,
        },
        {}
};

/* register the above sysctl */
if (!register_sysctl_table(sample_parent_table)) {
        printk(KERN_ALERT "Error: Failed to register sample_parent_table\n");
        return -EFAULT;
}
```

Note that the `ctl_name` value for the `sample_parent_table[]` is `CTL_KERN` which is defined in `<linux/sysctl.h>`. Since there is no predefined unique value (that can be used as its `ctl_name`) for the `sample_child_table[]`, `CTL_UNNUMBERED` must be used instead. `CTL_UNNUMBERED` allows to register sysctls without a binary number and that's probably what you want most of the time.

If you need a binary number assigned to your new sysctl, it must be defined in `<linux/sysctl.h>` first:

```
/* CTL_KERN names: */
enum
{
```

```
        KERN_OSTYPE=1,          /* string: system version */
        KERN_OSRELEASE=2,       /* string: system release */
        KERN_OSREV=3,           /* int: system revision */
        KERN_VERSION=4,         /* string: compile time info */
        KERN_SECUREMASK=5,      /* struct: maximum rights mask */
        ...
        KERN_SAMPLE=SOME_UNUSED_VALUE
};
```

Now `KERN_SAMPLE` can be used as `clt_name` for the `sample_child_table[]`:

```
static struct ctl_table sample_child_table[] = {
        {
                .ctl_name       = KERN_SAMPLE,
                .procname       = "sample",
                ...
```

Once the above sysctl is registered, it can used by sysctl utility:

```
> sysctl kernel.sample=3
```

This will set `global_var` to 3. Attempting to set a value that is < 0 or > 5 will produce the following error:

```
> sysctl kernel.sample=10
error: "Invalid argument" setting key "kernel.sample"
```