Rozdział 1

System plików PROC - temat nr. 66

W tym rozdziale zajmiemy się systemem plików znanym jako **PROC**, omawiając jego przeznaczenie oraz imlementację. Autorzy: Maciej Bagiński, Bartosz Biskupski.

1.1 Geneza

Już począwszy od jąder linii 1.0 w Linuksie wprowadzony został bardzo ciekawy i specyficzny zarazem system plików - tzw. PROC File System (zwany także **Linux-PROC**, albo krótko i konkretnie - **procfs**). Nazwa pochodzi od pierwotnego przeznaczenia systemu, jakim było wyłącznie umożliwienie użytkownikom łatwego dostępu do informacji o **PROC**esach. Z biegiem czasu okazało się jednak, że zastosowane w procfs podejście do systemu operacyjnego jest na tyle elastyczne i wygodne w użyciu, że warto rozszerzyć wachlarz oferowanych usług o inne operacje związane z działalnością systemu operacyjnego. Można powiedzieć, że z jądra na jądro, PROC ”rozkwita” i staje się coraz ważniejszym składnikiem Linuxa.

1.2 Wstęp

System plików PROC został zaimplementowany w ramach Linuksowego **VFS-a**[1] i posiada większość jego funkcjonalności. Jest on widziany w systemie plików jako zwykły i niepozorny katalog o nazwie */proc*. Procfs nie jest jednak zwykłym dyskowym systemem plików ! Mimo, że może być mount-owany jak zwykły system plików (typu proc), informacje o plikach przechowywane są nie na nośnikach danych, tylko w pamięci, a zawartość plików jest tworzona dynamicznie na żądanie użytkownika. Z tego też powodu nie ma sensu podawanie dla plików z procfs-a ich wielkości, podczas inicjowania ich logicznej reprezentacji w systemie plików, którą jest struktura `proc_dir_entry`, (bliżej zapoznamy się z nią później) ”narzucana” jest im najczęsciej wielkość zerowa (a czasami 64 bajty). Większość plików z katalogu */proc* może być czytana przez każdego użytkownika Linuksa, prezentując mu w ten sposób informacje o różnych częściach systemu operacyjnego, takie jak np. opisy procesów, charakterystyka i parametry jądra, informacje o urządzeniach (dyskach, kartach PCI, portach itp.), a nawet, o ile mamy do tego prawa, stan pamięci procesów

---

[1] VFS to tzw. *Virtual File System*, opisujący pliki z logicznego, a nie fizycznego punktu widzenia

bądź całego systemu !

Dostęp do plików procfs z punktu widzenia programisty nie różni się od dostępu do zwykłych plików tekstowych i pozwala na "ładną" implementację wielu programów, korzystających kiedyś, z konieczności, wprost ze struktur jądra. Dziś takie programy, których przykładem może być popularny "ps" czerpią potrzebne dane bez posiadania specjalnych uprawnień, po prostu odczytując je z plików procfs-a.

Obecny system procfs umożliwia także, w niektórych przypadkach, zapis do swoich plików. Co taka funkcjonalność może jednak oznaczać ? Otóż zapis do plików procfs-a umożliwia regulowanie parametrów systemu, zmianę polityki jego działania. Przykładowo, jeśli denerwuje nas, że po nacisnięciu ctrl-alt-delete musimy długo czekać na zamknięcie się systemu, to możemy zapisać do pliku */proc/sys/kernel/ctrl-alt-del* dowolną wartość większą od zera i jeśli wciśniemy teraz ctrl-alt-del, to nasz komputer zresetuje się bez całej długotrwałej procedury zamykania. Oczywiście nie jest to rozwiązanie zalecane, a zmiana nie jest trwała.

1.3 Opis wybranych plików z katalogu */proc*

Poniżej przedstawiony został opis ważniejszych plików z procfs. Bardziej szczegółowy znaleźć można w pliku proc.txt znajdującym się w dokumentacji źródeł jądra [2] 2.2.17. Nie wszystkie z wymienionych plików muszą znajdować się na konkretnym komputerze; zawartość katalogu */proc* zależy bowiem od konfiguracji: sprzętowej komputera i programowej - jądra Linuxa [3].

1.3.1 Katalog */proc*:

- **apm**

  zarządzanie energią (ang. *advanced power menagement*)
- **cmdline**

  parametry uruchomienia systemu
- **cpuinfo**

  szczegółowe informacje o procesorze
- **devices**

  lista dostępnych w systemie urządzeń znakowych i blokowych
- **dma**

  lista zarejestrowanych kanałów DMA
- **filesystems**

  lista obsługiwanych przez jądro systemów plików
- **interrupts**

  lista przerwań z opisami i liczbą wywołań

---

[2] źródła jądra najczęściej można znaleźć w katalogu */usr/src/linux*

[3] Konfigurację "programową" wybieramy podczas kompilacji jądra i ładowania jego modułów. Możemy na przykład zdecydować czy procfs ma w ogóle wchodzić w skład naszego systemu.

- **ioports**

  lista portów urządzeń wejścia / wyjścia
- **kcore**

  obraz pamięci jądra
- **kmsg**

  bufor z wiadomościami od jądra
- **ksyms**

  tablica symboli jądra
- **loadavg**

  statystyki pracy systemu
- **locks**

  blokady jądra
- **meminfo**

  szczegółowe informacje o pamięci operacyjnej i pamięci wymiany
- **modules**

  lista załadowanych modułów
- **mounts**

  lista zamountowanych systemów plików. Nasz ukochany proc też tu jest.
- **partitions**

  lista partycji na dysku widzianych przez system
- **pci**

  lista danych o podłączonych urządzeniach PCI
- **rtc**

  ustawienia dotyczące zegara
- **sound**

  informacja o sterowniku karty dźwiękowej
- **stat**

  statystyki jądra
- **swaps**

  informacja o użyciu pamięci wymiany

- **uptime**

  czas w sekundach od uruchomienia komputera oraz czas spędzony na ”nic nierobieniu”

- **version**

  informacje o jądrze systemu operacyjnego oraz o komputerze (wykorzystywane przez polecenie uname)

1.3.2 Katalogi */proc/<pid procesu>*:

Każdemu dziającemu w systemie procesowi odpowiada dokładnie jeden z tych katalogów.

- **cmdline**

  linia poleceń użyta do uruchomienia procesu

- **environ**

  wartości zmiennych środowiskowych procesu

- **exe**

  dowiązanie do kodu procesu

- **maps**

  informacja o zmapowanych obszarach pamięci procesu

- **mem**

  pamięć procesu. Uprawnione osoby mogą po niej pisać.

- **stat**

  statystyki procesu

- **statm**

  statystyki pamięci procesu, a w szczególności - rodzaje stron pamięci procesu

- **status**

  informacje o procesie

- **cwd**

  dowiązanie do bieżącego katalogu procesu

- **fd**

  katalog z wszystkimi deskryptorami plików procesu

- **root**

  dowiązanie do katalogu głównego procesu

1.3.3 Katalog */proc/ide*:

Zawiera pliki opisujące urządzenia IDE w komputerze, a w szczególności ich sterowniki oraz parametry fizyczne. Pliki są umieszczone w katalogach odpowiadających każdemu urządzeniu i każdej szynie ide.

1.3.4 Katalog */proc/net*:

Zawiera pliki informujące o działaniu sieci (np. plik **route** zawiera tablicę routingu naszego komputera) i protokołów sieciowych oraz katalog ip_masq zawierający pliki z tablicami maskaradowania pakietów różnych protokołów.

1.3.5 Katalog */proc/scsi*:

Tutaj znajdziemy informację o urządzeniach scsi, o ile mamy takie w komputerze.

1.3.6 Katalog */proc/self*:

Jest dowiązaniem do katalogu /proc/<pid ”czytacza”>, czyli katalogu procesu który usiłuje go odczytać.

1.3.7 Katalog */proc/sys/dev*:

Obecnie może zawierać jedynie podkatalog *cdrom*, w którym zawarte są parametry techniczne napędu cd.

1.3.8 Katalog */proc/sys/fs*:

Pliki określające parametry cache-owania katalogów, obsługi plików, i-node-ów i superbloków przez jądro.

1.3.9 Katalog */proc/sys/kernel*:

Ustawienia dotyczące zachowania się jądra w różnych sytuacjach (np. wspomniane wciśnięcie **ctrl-alt-del**) oraz identyfikacja komputera i systemu operacyjnego.

1.3.10 Katalog */proc/sys/net*:

Różne pliki odpowiedzialne za parametry socket-ów na różnych warstwach komunikacji, takich jak np. popularne Ethernet-IP-TCP, ustawienia router-owania, sterowanie zachowaniem protokołów.

1.3.11 Katalog */proc/sys/vm*:

Parametry potrzebne do zarządzania pamięcią i demonami ją obsługującymi (np. **bdflush**).

1.3.12 Katalog */proc/tty*:

Jego pliki informują o dostępnych bądź używanych tty.

1.4 Implementacja

1.4.1 Rejestrowanie systemu plików proc w systemie

Przy starcie systemu wykonywana jest funkcja `file_system_setup()`, która inicjuje systemy plików. Wywołuje ona `init_proc_fs()`, która z kolei przechodzi do `register_filesystem(&proc_fs_type)`. Ma ona za zadanie dodanie struktury typu `file_system_type`, podanej jako parametr, do globalnej listy `file_systems` wszystkich zarejestrowanych systemów plików. Po tej operacji procfs nadaje się już do mount-owania.

1.4.2 Montowanie systemu plików PROC

Funkcja `do_mount()` wywołuje `read_super()`. Ta wyszukuje na liście `file_systems` (lista zarejestrowanych systemów plików) system plików o nazwie "proc" i wykonuje wczytanie jego bloku specjalnego. Za to zadanie odpowiada funkcja `proc_read_super()`. Dla standardowych systemów plików zajmujących sie przechowywaniem danych na nośnikach (takich jak ext2, fat, etc.) odpowiednia funkcja wczytująca blok specjalny pobiera niezbędne informacje z dysku. Tutaj atrybuty struktury `super_block` są ustawiane statycznie. Ustawiane są operacje dla bloku specjalnego, przydzielany jest `root_inode` procfs-a, ustawiane są dla niego operacje (czyli struktura `proc_root_inode_operations`) oraz przetwarzane dane związane z mount-owaniem (np. można ustawić właściciela `root_inode`).

1.4.3 Drzewo plików procfs

Podstawową strukturą, zawierającą atrybuty plików jest:

```
struct proc_dir_entry {
/* Numer i-węzła */
unsigned short low_ino;

/* Długość nazwy pliku */
unsigned short namelen;

/* Nazwa pliku */
const char *name;

/* Atrybuty pliku */
mode_t mode;

/* Liczba sztywnych łączy (plików) w katalogu (dla katalogu) */
nlink_t nlink;

/* Właściciel */
uid_t uid;

/* Grupa właściciela */
gid_t gid;
```

```
/* Rozmiar */
unsigned long size;

/* Operacje na inode-dzie */
struct inode_operations * ops;

/* Funkcja wywoływana przy odczycie z pliku (zwykle jest to NULL,
korzysta się wyłącznie z pola ops proc_dir_entry) */
int (*get_info)(char *, char **, off_t, int, int);

/* Funkcja ustawiająca w i-węźle atrybuty pliku */
void (*fill_inode)(struct inode *, int);

/* Wskaźnik do struktury proc_dir_entry opisującej następny plik w tym samym katalogu */
struct proc_dir_entry *next;

/* Wskaźnik do katalogu nadrzędnego pliku */
struct proc_dir_entry *parent;

/* Wskaźnik do pierwszego pliku w katalogu (o ile nasza struktura
reprezentuje katalog, bo jeśli nie to NULL) */
struct proc_dir_entry *subdir;

/* Dane */
void *data;

/* Funkcje realizujące różne zadania związane z plikiem (zwykle
wywoływane, gdy brakuje pola ops) */
int (*read_proc)(char *page, char **start, off_t off,
 int count, int *eof, void *data);
int (*write_proc)(struct file *file, const char *buffer,
  unsigned long count, void *data);
int (*readlink_proc)(struct proc_dir_entry *de, char *page);

/* Licznik użyć */
unsigned int count;

/* Flaga usunięcia */
int deleted;

};
```

System plików proc składa się z drzewa takich struktur powiązanych ze sobą wskaźnikami w polach next, parent i subdir. Dostęp do struktury `proc_dir_entry` danego pliku można uzyskać poprzez dodatkowe pole wprowadzone w i-węźle plików procfs. Znajduje się ono w unii ‘u’ i nosi

Rysunek 1.1: Przykładowe drzewo struktur `proc_dir_entry` ([Projekty z lat ubiegłych])

nazwę `generic_ip`.

Struktury `proc_dir_entry` są zadeklarowane statycznie dla głównego katalogu procfs oraz plików w katalogach procesów. Dla innych katalogów i plików procfs, takich jak np. */proc/sys/** zdefiniowane są lekko okrojone struktury, które podczas inicjowania zamieniane są na właściwe `proc_dir_entry`. Struktury są łączone w drzewo za pomocą funkcji `proc_register()` mającej swoje źródło w *<źródła jądra>/fs/proc/root.c*

Dla wszystkich katalogów procesów zadeklarowana jest tylko jedna struktura `proc_dir_entry`, natomiast osobne struktury tworzone są w pamięci dynamicznie podczas wołania funkcji lookup() interfejsu VFS dla root-inode (funkcja `proc_root_lookup()`).

Struktury `proc_dir_entry` wykorzystywane są do uzupełnienia i-węzłów niezbędnymi informacjami, takimi jak prawa dostępu, właściciel, rozmiar, operacje na inode-dzie. I-węzeł jest tworzony podczas wykonywania funkcji `proc_lookup()` (a dokładniej w `proc_get_inode()`) odpowiadającej funkcji `lookup()` standardowego interfejsu VFS.

Numery i-węzłów dla większości plików są z góry ustalone i pozostają zawsze takie same. Wyjątkiem są katalogi procesów wraz z plikami znajdującymi się wewnątrz nich. Ich numery są kombinacją ich PID-u oraz rodzaju pliku (dzieki temu możemy określić na podstawie i-węzła proces którego on dotyczy). Szczegółowo sposób numeracji i-węzłów jest opisany w dokumencie *inode-alloc.txt* położonym w katalogu *<źródła jądra>/fs/proc/*

Większość budowanych przez procfs i-węzłów ma niewielką liczbę udostępnianych operacji dla interfejsu VFS. Zwykle są to tylko `lookup()` i `readdir()` dla katalogów i `read()` dla plików (ale np. niektóre pliki w katalogu */proc/sys/* mają również zdefiniowane funkcje zapisu, umożliwiając modyfikowanie wybranych parametrów jądra, a pliki mem procesu umożliwiają zapis do pamięci konkretnego procesu).

1.4.4 Odczytywanie pliku z katalogu */proc*

Podczas montowania systemu plików proc w pamięci został stworzony root i-node. Korzystając z informacji w nim zawartych możemy odnaleźć szukany przez nas plik o danej nazwie w procfs. W tym celu VFS wywołuje funkcję lookup() dla root-inode, którą jest w procfs `proc_root_lookup()`. Wywołuje ona funkcję `proc_lookup()`, która wyszukuje wśród synów wierzchołka poddrzewa struktur `proc_dir_entry` zapisanego w polu `u.generic_ip` pozycję, której nazwa pliku zgadza się z szukaną przez nas. Jeżeli jej nie znajdzie, poszukiwania kontynuuje funkcja `proc_root_lookup()`. Sprawdza ona, czy nazwa ta przypadkiem nie odpowiada PID-owi jakiegoś procesu w systemie. Znaleziony i-węzeł jest uzupełniany o informacje (np. adres pola `i_op`) z odpowiadającej mu struktury `proc_dir_entry` (funkcja `proc_get_inode()`), a następnie jest dodawany do struktury `dentry` katalogu. W tej chwili można wykonywać na nim operacje VFS takie jak odczytanie zawartości, etc, a funkcje je wykonujące zostały zdefiniowane w strukturze `proc_dir_entry` pliku).

Podobnie przebiega znajdowanie pliku/katalogu w którymś z podkatalogów */proc* - pomijana jest wtedy funkcja `proc_root_lookup()` i od razu wywoływane jest `proc_lookup()`.

Odczytem zawartości plików z katalogu głównego procfs oraz plików w katalogach procesów zajmuje się funkcja `array_read()` (tak zostało ustawione w strukturach `proc_dir_entry` tych plików). Alokuje ona pamięć na dane wyjściowe a następnie próbuje wykonać funkcję wskazywaną przez `get_info` w `proc_dir_entry`. Jeśli jest ona ustawiona na NULL, to w zależności od tego, czy jest to i-węzeł pliku w katalogu jakiegoś procesu, czy plik w katalogu głównym procfs, wywołuje odpowiednio `get_process_array()` albo `get_root_array()`. Obie te funkcje pobierają z numeru i-węzła typ pliku z jakim mają do czynienia, a następnie wywołują odpowiednią dla niego funkcję odczytującą zawartość.

1.4.5 Jak dodać nowy plik do katalogu */proc* ?

1. Należy dodać nową wartość enum w pliku *<źródła jądra>/include/linux/proc_fs.h*

2. Dodać nową procedurę, która będzie zapisywać wartość w postaci tekstowej pod wskazany adres i zwracać liczbę zapisanych bajtów do pliku *<źródła jądra>/fs/proc/array.c*, bo tam właśnie znajdują się funkcje wykonujące "brudną robotę":

   int `moja_procedura ( char* bufor )`

   UWAGA: bufor to jedna strona!

3. dodać wywołanie `moja_procedura` do `get_root_array()`

   (znów w pliku *<źródła jądra>/fs/proc/array.c*)

4. zarejestrować nowy plik w `proc_root_init()`, poprzez utworzenie dla niego nowej stuktury `proc_dir_entry` oraz jej zarejestrowanie funkcją `proc_register`

   (plik *<źródła jądra>/fs/proc/root.c*)

1.5 Opis wybranych plików źródłowych

- *<źródła jądra>/include/linux/proc_fs.h*:

Definicje numerów węzłów (i-node) dla poszczególnych plików; część procedur rejestrujących zdefiniowanych 'inline' - net i scsi.

- *<źródła jądra>/fs/proc*:
  - *array.c*:

    Definicje procedur pobierających zarówno dla katalogu głównego, jak i dla katalogów procesów; operacje na pojedyńczych węzłach w katalogu procesu;
  - *base.c*:

    Zawiera procedury rejestrujące pliki w katalogu procesu, tj. implementację readdir dla katalogów procesów. Zawiera także poszczególne struktury `proc_dir_entry`;
  - *fd.c*:

    Funkcje implementujące `lookup` i `read` dla katalogów */proc/<pid>/fd*
  - *generic.c*:

    Zawiera domyślne funkcje i struktury operacji na i-węzłach, a także funkcję `create_proc_dir_entry`, tworzącą nowy węzeł
  - *inode-alloc.txt*:

    Opis przydziału numerów i-węzłów poszczególnym typom plików procfs
  - *inode.c*:

    Funkcje wczytujące super blok oraz tworzące i-węzły na podstawie struktur `proc_dir_entry`.
  - *kmesg.c*:

    Funkcje implementujące zachowanie się pliku */proc/kmsg*
  - *mem.c*:

    Funkcje implementujące operacje wykonywane na pliku */proc/<pid>/mem*
  - *net.c*:

    Implementacja funkcji czytania pliku dla wszystkich plików z */proc/net/* Ta funkcja jest "uszczegóławiana" w plikach z katalogu *<źródła jądra>/net*
  - *proc_tty.c*:

    Funkcje rejestrujące dla wszystkich plików i katalogów z */proc/tty* oraz funkcje odczytu dla plików */proc/tty/drivers* i */proc/tty/ldiscs*
  - *procfs_syms.c*:

    Procedura rejestrująca */proc* w VFS
  - *root.c*:

    Rejestracja plików katalogu */proc*; procedury przeglądania katalogu i implementacja linku *self* ; operacje na plikach w głównym katalogu
  - *scsi.c*:

    Domyślne operacje zapisu i odczytu wykonywane na plikach z */proc/scsi*

- *<źródła jądra>/kernel*:

  - *sysctl.c*

    Zawiera funkcje odpowiedzialne za działanie plików z katalogu */proc/sys*, najczęściej z możliwoscią zapisu !

- *<źródła jądra>/include/linux*:
  - *sysctl.h*

    Struktury danych i funkcje potrzebne w *<źródła jądra>/kernel/sysctl.c*

- *<źródła jądra>/net* i podkatalogi :

  Różne pliki *.c, które zawierają definicje funkcji czytających dla proc-a i rejestrujących poszczególne pliki z katalogu */proc/net*

- *<źródła jądra>/drivers* i podkatalogi :

  różne pliki *.c, które zawierają definicje funkcji czytających dla proc-a i rejestrujących poszczególne pliki z katalogu */proc/ide* oraz */proc/scsi*

1.6 Zadania

1. Spróbuj dodać nowy plik o dowolnej funkcji i dowolnej ścieżce do katalogu /proc, nie wykonując rekompilacji jądra.

1.7 Rozwiązania zadań

1. Oto co należy zrobić:
   (a) Utworzyć w pamięci strukturę `file_operations` z ustawionymi polami operacji na pliku, które chcemy udostępniać (najczęściej będzie to read)
   (b) Utworzyć w pamięci strukturę `inode_operations` z adresem poprzednio utworzonej struktury w polu `default_file_operations`
   (c) Utworzyć w pamięci nową strukturę `proc_dir_entry` z adresem poprzednio utworzonej struktury w polu ops, wybraną nazwą pliku i jej długością, unikalnym numerem i-węzła, atrybutami pliku.
   (d) Połączyć ją z drzewem pozostałych struktur (można do tego wykorzystać funkcję `proc_register()`). Adres korzenia tego drzewa znajduje się w polu `u.generic_ip` głównego i-węzła procfs.

Literatura

[Źródła jądra] Źródła jądra 2.2.17, zwłaszcza te z katalogu *<źródła jądra>/fs/proc*.

[Projekty z lat ubiegłych] Projekty przygotowywane w ramach zajęć z Systemów Operacyjnych przez studentów MIMUW w ubiegłych latach.

[Bach 95] Maurice J. Bach: *Budowa systemu operacyjnego UNIX* Warszawa, WNT 1995.

[SO lab] Zajęcia laboratoryjne z Systemów Operacyjnych na temat procfs.