

Systemy Operacyjne 1
Laboratorium 8
„Obsługa terminali”
(jeden tydzień)

dr inż. Arkadiusz Chrobot

27 listopada 2020

Wstęp

Instrukcja zawiera informacje na temat niskopoziomowej obsługi terminali w systemie Linux. Pierwszy rozdział przedstawia ogólnie koncepcję terminali. W drugim rozdziale opisano API terminali. Instrukcja kończy się listą zadań do samodzielnego wykonania.

1. Terminale

Każdy system komputerowy jest wyposażony w pewną liczbę urządzeń przeznaczonych do komunikacji z użytkownikiem lub innymi komputerami. Mianem „terminal” określa się najczęściej monitor i klawiaturę, choć można on również oznaczać takie urządzenia jak modem i drukarka. W przypadku systemów uniksowych określa się tak również protokół wyznaczający sposób obsługi danego urządzenia przez system operacyjny i część systemu obsługującą urządzenia do interakcji z użytkownikiem. Istnieje zbiór funkcji wchodzących w skład API systemu Linux, które umożliwiają programom użytkownika sterowanie zachowaniem terminali. Z każdym terminalem są związane dwie kolejki, wejściowa i wyjściowa, które stanowią rodzaj buforów jądra na znaki, które mają zostać zapisane na ekran, bądź które zostały pobrane z klawiatury. Każdy terminal może pracować w dwóch trybach: kanonicznym i niekanonicznym. W pierwszym trybie znaki przeczytane lub zapisywane do terminala są przetwarzane grupami, które kończą znaki nowego wiersza, końca wiersza lub końca pliku. Ponadto niektóre znaki mają specjalne znaczenie, np.: mogą powodować skasowanie poprzedzającego je znaku. W trybie niekanonicznym program może określić samodzielnie jakimi porcjami mają być czytane i zapisywane dane do terminala, oraz które znaki będą znakami sterującymi.

2. API terminali

Podstawową strukturą danych wykorzystywaną w obsłudze terminali jest struktura `termios`. Standardy uniksowe gwarantują, że będą się w niej znajdować przynajmniej następujące pola:

`c_iflag` - pole określające flagi trybu wejścia,

`c_oflag` - pole określające flagi trybu wyjścia,

`c_cflag` - pole określające flagi trybu sterowania,

`c_lflag` - pole określające flagi trybu lokalnego,

`cc_t c_cc[NCCS]` - tablica określająca, które znaki są związane z funkcjami sterującymi.

Uwaga:

Niektóre terminale mogą definiować własne pola w tej strukturze, związane np. z obsługą kolorowych wyświetlaczy. Dlatego przy manipulacji ustawieniami terminala należy najpierw przeczytać jego bieżące ustawienia i zapamiętać je w osobnej zmiennej. Następnie można zmodyfikować odpowiednie pole lub pola struktury `termios` przy pomocy odpowiednich stałych i operatorów bitowych, a następnie wywołać funkcję zmieniającą zachowanie terminala. Przed zakończeniem program powinien przywrócić domyślną obsługę terminala (domyślne ustawienia).

Do zarządzania ustawieniami terminala służą następujące funkcje i makra:

`isatty()` - funkcja sprawdza, czy podany jej jako argument wywołania deskryptor pliku (zmienna typu `int`) związany jest z plikiem urządzenia terminala. Jeśli tak jest, to zwraca 1, a jeśli nie to 0.
Szczegóły: `man isatty`

`ttyname()` - funkcja zwraca wskaźnik na łańcuch znaków będący nazwą terminala powiązanego z przekazanym jej jako argumenty wywołania deskryptorem, lub `NULL` w przypadku błędu. Wskazywany łańcuch może być nadpisany przez wywołania kolejnych funkcji.
Szczegóły: `man ttyname`

ttynam_r() - funkcja przyjmuje trzy argumenty wywołania. Pierwszym jest deskryptor związany z terminalem, drugim wskaźnik na tablicę znaków, a trzecim rozmiar tej tablicy. W drugim argumencie zapisywana jest nazwa terminala związanego z deskrytorem. Jeśli ta operacja się powiedzie, to funkcja zwróci wartość 0, w przeciwnym przypadku wartość różną od zera.

Szczegóły: `man ttynam_r`

tcgetattr() - funkcja przyjmuje dwa argumenty wywołania: deskryptor pliku związanego z terminalem oraz wskaźnik na strukturę `termios` i zapisuje ustawienia terminala związanego z deskrytorem do tej struktury. Jeśli działanie funkcji się powiedzie, to zwraca ona wartość 0, a w przeciwnym razie -1.

Szczegóły: `man tcsetattr`

tcsetattr() - funkcja służy do zmiany ustawień terminala. Przyjmuje trzy argumenty wywołania: deskryptor pliku związanego z terminalem, stałą określającą kiedy zmiana nastąpi oraz wskaźnik na strukturę `termion`, zawierającą nowe ustawienia dla terminala. Drugi argument może być następujący:

TCSANOW - zmiana będzie wykonana natychmiast,

TCSADRAIN -zmiana nastąpi, kiedy wszystkie dane wyjściowe zapisane do pliku terminala zostaną przetworzone. Ta stała powinna być używana tylko wtedy, gdy zmiany mają wpływ na wyjście terminala.

TCSAFLUSH - zmiany zostaną dokonane po przetworzeniu danych wyjściowych, wszystkie nieprzetworzone dane wejściowe zostaną usunięte.

Funkcja zwraca 0 jeśli zmiana się powiedzie, a -1 w przeciwnym przypadku.

Szczegóły: `man tcsetattr`

VMIN - stała (makro), używana w konfigurowaniu trybu niekanonicznego. Określa element tablicy `c_cc` (pole struktury `termios`), który definiuje ile bajtów musi znaleźć się w kolejce wejściowej, zanim zostanie zakończona operacja odczytu.

VTIME - stała (marko) używana w konfigurowaniu trybu niekanonicznego. Określa element tablicy `c_cc` (pole struktury `termios`), który definiuje jak długo będzie czekał system, zanim zakończy operację wejścia. Jednostką czasu jest 0,1 sekundy.

Szczegółowy opis wszystkich funkcji i struktur danych związanych z obsługą terminali znajduje się w podręczniku systemowym (`man termios`) i w dokumentacji biblioteki `glibc`: *The GNU C Library Reference Manual* autorstwa Sandry Loosemore, Richarda Stallmana, Rolanda McGratha, Andrew Orama i Urlicha Dreppera.

Zadania

UWAGA: PROGRAMY MUSZĄ BYĆ NAPISANE Z PODZIAŁEM NA FUNKCJE Z PARAMETRAMI ORAZ MUSZĄ SPRAWDZAĆ, CZY WYWOŁYWANE PRZEZ NIE FUNKCJE Z API SYSTEMU OPERACYJNEGO NIE SYGNALIZUJĄ WYJĄTKÓW.

1. Napisz program, który sprawdzi, czy z deskrytorem standardowego wejścia (`STDIN_FILENO` lub wartość 0) jest związany terminal. Jeśli tak, to program powinien wypisać nazwę pliku urządzenia tego terminala, oraz kilka jego bieżących ustawień.
2. Napisz program, który wyłączy echo, czyli wypisywanie znaków wprowadzonych z klawiatury na ekran. Po ponownym uruchomieniu program powinien przywrócić echo.
3. Napisz program demonstrujący obsługę terminala w trybie niekanonicznym (Może to być np. program posiadający menu, które jest uaktywnione naciśnięciem klawisza, bez konieczności potwierdzenia klawiszem `Enter`).
4. Napisz program, po którego uruchomieniu terminal będzie wypisywał wszystkie informacje dużymi literami (niezależnie od ustawienia `CapsLock`). Efekt ten powinno usuwać ponowne uruchomienie programu.