Procedury ekranowe systemu Unix

Witold Paluszyński
witold@ict.pwr.wroc.pl
http://sequoia.ict.pwr.wroc.pl/~witold/

Niniejszy dokument zawiera materiały do wykładu na temat wykorzystania biblioteki curses systemu Unix w Pascalu. Jest on udostępniony pod warunkiem wykorzystania wyłącznie do własnych, prywatnych potrzeb i może być kopiowany wyłącznie w całości, razem z niniejszą stroną tytułową.

Terminale znakowe w systemie Unix

Terminal znakowy jest istotnym elementem architektury systemu Unix. Jakkolwiek coraz rzadziej są już używane rzeczywiste, fizyczne terminale komunikujące się z systemem Unix asynchroniczną transmisją znakową, wirtualna wersja takiego terminala jest tworzona w systemie dla procesów jako element realizacji standardowych strumieni wejścia/wyjścia INPUT i OUTPUT. Przydatna jest również taka koncepcja terminala znakowego przy połączeniach sieciowych z komputerem Unixowym.

Działanie tych terminali polega na przesyłaniu i odbieraniu znaków jeden po drugim (szeregowo). Znaki pisane przez użytkownika na klawiaturze terminale wysyłają komputerowi, natomiast znaki przysyłane im przez komputer terminale wyświetlają na ekranie użytkownika, zapewniając dodatkowo np. przesuwanie kolejnych linii na ekranie (tzw. operacja *scroll*).

Terminale znakowe i operacje ekranowe

Poza ściśle szeregową transmisją znaków terminale posiadają zdolność wykonywania prostych operacji pełnoekranowych, np. gaszenie ekranu, wyświetlanie danych w określonym miejscu ekranu, podświetlanie, itp. Jednak użycie tych operacji w programie jest skomplikowane, ze względu na dużą liczbę typów terminali i ich zróżnicowanie. Dlatego powstała metoda obsługi terminali znakowych, której sens można streścić jako:

standaryzacja — wielość typów terminali i zróżnicowanie ich własności wymaga rozwiązania systemowego, ktorym może być poziom pośredni — schemat lub język opisu własności terminala

uproszczenie — aby nie musieć wykonywać wszystkich operacji na poziomie własności terminali wygodna byłaby biblioteka procedur realizujących typowe operacje ekranowe dla dowolnego terminala

przenośność — między różnymi systemami, którą uzyskujemy w efekcie

optymalizacja — często chcemy maksymalnie skrócić ilość danych przesyłanych do terminala ze względu na niewielką prędkość transmisji

Baza danych terminfo — opis własności terminala

```
z29|zenith29|z29b|,
    am, msgr,
    cols#80, lines#24,
    bel=^G, cbt=\E-, clear=\EE$<14>, cnorm=\Ey4, cr=\r,
    ... (ciag dalszy patrz skrypt lub ‘infocmp z29‘)
```

- własności binarne terminala, np.: „am” (*automatic margins*), „msgr”
- własności o wartościach liczbowych, np. liczby wierszy i kolumn danych
- własności o wartościach znakowych, zwane komendami terminala, np. „clear” oznacza zdolność wyczyszczenia ekranu terminala sekwencją ESCAPE+„E”, plus 14 milisekund opóźnienia

własności, których terminal nie posiada można zastąpić sekwencjami innych komend; za minimalną uznaje się zdolność do bezpośredniego adresowania ekranu, tzn. umieszczania kursora w dowolnym miejscu ekranu

Biblioteka curses — wyższy poziom abstrakcji

Programy ekranowe mogą korzystać z bazy terminfo bezpośrednio i niektóre tak rzeczywiście robią, np. edytor Gnu Emacs stara się działać sprawnie i wykorzystać do maksimum wszystkie możliwości terminala. Jeśli terminal nie posiada jakiejś komendy, to program wykonuje odpowiednią operację innymi komendami. W ostateczności każdy efekt można osiągnąć przy pomocy bezpośredniego adresowania ekranu.

Pisanie programów w ten sposób nie jest jednak łatwe. Wymaga bowiem wielowarunkowego wykonywania każdej najprostszej operacji. Tymczasem do budowy programów z interfacem ekranowym potrzebne są pewne proste i bardzo schematyczne konstrukcje ekranowe.

Rozwiązaniem jest biblioteka typowych procedur ekranowych napisana z wykorzystaniem własności terminfo, a więc w pełni uniwersalnych i działających poprawnie dla wszystkich terminali, a jednocześnie zapewniających programiście izolację od tych szczegółów. Biblioteka curses jest systemową biblioteką Unixa, dostępną we wszystkich wersjach systemu i zapewniającą tzw. źródłową przenośność programów.

Biblioteka curses — podstawowe pojęcia

okna: prostokątne obszary na ekranie, z którymi może być związana pewna treść w nich wyświetlana, oraz pewne operacje; linie i kolumny tekstu liczone są od zera poczynając od lewego górnego rogu

okno główne: cały obszar ekranu, który jest domyślnym oknem dla wyświetlanego tekstu (jedyne okno, którego nie trzeba tworzyć jawnie)

kursor okna: niewidzialna pozycja pamiętana dla każdego okna, w którym domyślnie może być wyświetlany tekst (kursor okna można przemieszczać funkcjami curses)

kursor terminala: widoczny na ekranie pojedynczy kursor pojawiający się w miejscu bieżącego wyświetlanego tekstu

atrybuty znaków: cechy wyświetlanych znaków takie jak: podświetlenie, pogrubienie, podkreślenie, miganie tekstu, itp.; różne terminale mogą mieć różne zestawy atrybutów (lub nie mieć ich w ogóle)

klawisze funkcyjne: dodatkowe klawisze terminala, np.: strzałki, HOME, END, F1, F2, ..., itd.; klawisze te wysyłają określone sekwencje znaków

Biblioteka curses — podstawowe zasady użycia

Biblioteka curses wymaga inicjalizacji, w celu utworzenia jej własnych struktur, jak również w celu przygotowania terminala do pracy pełnoekranowej. Przed pierwszym użyciem jakiejkolwiek funkcji z tej biblioteki konieczne jest wywołanie funkcji `INITSCR`, a po zakończeniu pracy w trybie curses należy wywołać procedurę `ENDWIN`.

Żadne teksty nie są wyświetlane na ekranie w chwili wywołania samych procedur wyświetlania (takich jak `WADDSTR`). Te procedury jedynie umieszczają teksty w specjalnym buforze ekranu, a wypisanie zawartości bufora **danego okna** na ekran, z optymalizacją, dokonuje się po wywołaniu procedury `WREFRESH`, np. po wykonaniu całej serii operacji wyświetlania.

W trakcie pracy w trybie curses nie można wykonywać żadnych operacji wejścia/wyjścia na terminalu metodami podstawowymi, tzn. pisząc i czytając na standardowych plikach tekstowych INPUT i OUTPUT (operacje na plikach dyskowych są dozwolone). Wszelkie próby wykonania takich operacji, jakkolwiek mogą dać pewne częściowe wyniki, naruszają logiczną organizację biblioteki curses, i uniemożliwią jej poprawna pracę!

Biblioteka curses — przykład wyświetlania

```
VAR ekran: WINDOW;

{...}
ekran := INITSCR;
WMOVE(ekran, LINES DIV 2 - 1, COLS DIV 2 - 12);
WADDSTR(ekran, 'Srodek                            ');
WREFRESH(ekran);
WADDSTR(ekran, 'ekranu                            ');
WREFRESH(ekran);
ENDWIN;
```

Program ilustruje inicjalizację biblioteki, przesunięcie kursora do punktu o danych współrzędnych, a potem wpisanie tekstu do okna głównego i wyświetlenie jego zawartości. Stałe `LINES` i `COLS` oznaczają liczbę linii i kolumn głównego okna, a zarazem całego ekranu terminala.

Inne procedury służące do umieszczania tekstu w oknach: `WADDSTR` (napisy znakowe), `WADDCH` (pojedyncze znaki), `WPRINTWINT` (liczby całkowite). Uogólnione wersje tych procedur wykonują jednocześnie przesunięcie kursora i wyświetlanie: `MVWADDSTR`, `MVWADDCH`, oraz `MVWPRINTWINT`.

Biblioteka curses — przykład czytania znaków

Czytanie znaków jest zawsze związane z jednym konkretnym oknem. Służą do tego dwie procedury: `WGETCH` i `WGETSTR`, odpowiednio do przeczytania jednego znaku i całego łańcucha znaków.

```
ekran := INITSCR;
CBREAK;
NOECHO;
WADDSTR(ekran, 'Nacisnij dowolny klawisz                ');
WREFRESH(ekran);
z := WGETCH(ekran);
WADDSTR(ekran, 'Wprowadzony zostal znak "               ');
WADDCH(ekran, chr(z));
WADDSTR(ekran, '"                                       ');
WREFRESH(ekran);
```

Pojedyncze znaki czytane funkcją `WGETCH` otrzymujemy jako liczbowe kody znaków.

Przy czytaniu znaków w trybie ekranowym przydatne są specjalne tryby pracy drivera terminala systemu Unix, czyli jego części odpowiadającej z transmisję znaków z i na terminal. Jednym z takich trybów jest tzw. tryb *cbreak* (procedury `CBREAK` i `NOCBREAK`), polegający na wyłączeniu buforowania całej linii wprowadzanego tekstu, tzn. każdy pojedynczy znak może być przeczytany natychmiast po naciśnięciu klawisza na klawiaturze.

W tym trybie nie ma potrzeby naciskania klawisza ENTER po wprowadzeniu linii tekstu, ale nie jest też oczywiście możliwe np. skasowanie pomyłkowo wpisanego znaku i wpisanie innego — pierwszy znak został już przeczytany przez program.

Innym trybem pracy drivera terminala, który również można kontrolować procedurami biblioteki curses, jest tryb *echa*, polegający na wyświetlaniu na terminalu danych wprowadzanych przez użytkownika z klawiatury. Przy czytaniu znaków w trybie okienkowym echo jest często niepotrzebne i można je wyłączyć procedurą `NOECHO`. Wtedy znak napisany na klawiaturze nie pojawia się w ogóle na ekranie.

Działanie procedur `CBREAK` i `NOECHO` jest niezależne, ale w praktyce tryb *cbreak* wygodnie jest używać łącznie z trybem *noecho*.

Biblioteka curses — użycie atrybutów wyświetlania

Niektóre terminale mają możliwość wyświetlania tekstów odmienną czcionką (np. pogrubioną), lub z innymi cechami charakterystycznymi, takimi jak mruganie. W poniższym przykładzie widać połączenie tekstu wyświetlanego normalnie z fragmentem wyświetlanym w tak zwanym rewersie (ang. *reverse video*), to znaczy ciemne litery na jasnym tle.

```
ekran := INITSCR;
WMOVE(ekran,4,0);
WADDSTR(ekran, 'Prosze nacisnac jakis                   ');
WMOVE(ekran,4,21);
WATTRON(ekran, A_REVERSE);
WADDSTR(ekran, 'klawisz funkcyjny                       ');
WATTROFF(ekran, A_REVERSE);
WMOVE(ekran,5,0);
WREFRESH(ekran);
ENDWIN;
```


Biblioteka curses — czytanie klawiszy funkcyjnych

Niektóre klawisze terminala, takie jak strzałki, klawisze F1, F2, itp., zwane klawiszami funkcyjnymi, są specjalne bo nie wysyłają żadnych konkretnych znaków kodu ASCII, lecz sekwencje znaków, zależne od typu terminala.

Procedura `WGETCH` potrafi rozpoznać takie sekwencje jako odpowiadające klawiszom funkcyjnym, i zwraca wartość pojedynczego znaku. Ponieważ jednak nie jest to prawdziwy znak, zatem zwrócona wartość nie jest kodem znaku, i może być rozpoznana jedynie przez porównanie ze specjalnie zdefiniowanymi w bibliotece curses stałymi, jak widać w przykładzie.

```
ekran := INITSCR;
CASE WGETCH(ekran) OF
  KEY_DOWN  : {strzalka w dol};
  KEY_UP    : {strzalka w gore};
  KEY_LEFT  : {strzalka w lewo};
  KEY_RIGHT : {strzalka w prawo};
  KEY_PF1   : {klawisz funkcyjny PF1};
  {... i inne}
  END;
ENDWIN;
```

Biblioteka curses — kasowanie tekstu w oknie

```
ekran := INITSCR;
WMOVE(ekran,2,0);
WADDSTR(ekran, 'Nacisnij <RETURN> by zmazac stad do konca');
WMOVE(ekran,3,0);
WADDSTR(ekran, 'Ta linia tez zostanie zmazana.          ');
WMOVE(ekran,2,31);
WREFRESH(ekran);
z := WGETCH(ekran);
WCLRTOBOT(ekran);
WREFRESH(ekran);
ENDWIN;
```

Po wpisaniu kilku linijek tekstu i wykonaniu operacji WREFRESH, program czyta jeden znak z klawiatury, przez co czeka na reakcję użytkownika. Następnie przez wywołanie procedury WCLRTOBOT kasuje całą zawartość okna od wskazanego miejsca do końca, tzn. do najniższej linijki włącznie. Podobna procedura WCLRTOEOL kasuje tekst do końca danej linijki, a WCLEAR całą zawartość okna.

Biblioteka curses — tworzenie okien dodatkowych

W przykładzie ekran główny służy do wyświetlania wyników pewnych poleceń użytkownika, wpisywanych w małym okienku dialogowym na dole ekranu, które włącza się tylko w chwili naciśnięcia przez użytkownika klawisza „c". Normalnie program pracuje w trybie NOECHO, a jedynie na czas wpisywania komendy w okno dialogowe włączany jest tryb ECHO. Cały czas natomiast włączony jest tryb CBREAK.

```
VAR bufor: STRING80;
    ekran, dialog, i, z: INTEGER;

ekran := INITSCR;
NONL; NOECHO; CBREAK;
dialog := NEWWIN(3,COLS-10,LINES-5,5);{male ok.z dolu ekr}
FOR i := 0 TO LINES-1 DO          {zapelniamy okno glowne}
  BEGIN
    MVWADDSTR(ekran,i,0,
            'To jest linijka ekranu numer          ');
    WPRINTWINT(ekran,i);
  END;
```



```
WHILE TRUE DO        {pomijamy kwestie zakonczenia endwin}
  BEGIN
    WREFRESH(ekran);
    z := WGETCH(ekran);
    CASE CHR(z) OF
      'c': BEGIN
             WERASE(dialog);
             ECHO;
             BOX(dialog,0,0);
             WMOVE(dialog,1,1);
             WADDSTR(dialog,
              ' Podaj komende:                          ');
             TOUCHWIN(dialog);
             WREFRESH(dialog);
             WGETSTR(dialog,bufor);
             TOUCHWIN(ekran);
             NOECHO;
             { ... tu wykonujemy wczytana komende }
           END;
      'q': BEGIN
             ENDWIN;
             EXIT(0);
           END;
      OTHERWISE FLASH;
      END; {CASE}
  END;
```


Na uwagę w tym programie zasługuje organizacja wyświetlania obu okien. W chwili uaktywniania okna dialogowego wywoływana jest procedura TOUCHWIN, która powoduje pełne wyświetlenie tego okna (wyłączenie optymalizacji wyświetlania) przy kolejnym wywołaniu procedury WREFRESH.

Bez tego teksty mogłyby być nieprawidłowo wyświetlane, ponieważ biblioteka curses wyświetla treść każdego okna niezależnie, i nie ingeruje w to, co jest wyświetlane we wspólnych obszarach dwóch okien. Użycie procedury TOUCHWIN, wyłączając optymalizację wyświetlania, powoduje wyświetlenie całej zawartości danego okna.

Biblioteka curses — kompilacja kompilatorem HP Pascal

```
CONST $INCLUDE '/usr/local/air/hppc/include/pcurses.cons';$

TYPE  $INCLUDE '/usr/local/air/hppc/include/pcurses.type';$

VAR
   napis       : string80;
   ekran       : WINDOW;

$INCLUDE '/usr/local/air/hppc/include/pcurses.proc';$

{...}

   napis := 'dowolny napis';
   arrayTOstring(napis);
   WADDSTR(ekran, napis);
```

w celu skompilowania programu:

```
pc program.p /usr/local/air/hppc/lib/pcurses.o -lcurses
```


Biblioteka curses — kompilacja kompilatorem Sun Pascal

```
CONST
#include "/usr/local/air/sunpc/include/pcurses.cons"

TYPE
#include "/usr/local/air/sunpc/include/pcurses.type"

VAR
   ekran : WINDOW;
   napis : string80;

#include "/usr/local/air/sunpc/include/pcurses.proc"

{...}

   napis := 'dowolny napis';
   arrayTOstring(napis);
   WADDSTR(ekran, napis);
```

w celu skompilowania programu:

```
pc -L program.p /usr/local/air/sunpc/lib/pcurses.o -lcurses
```


Biblioteka curses — kompilacja kompilatorem GNU Pascal

```
CONST
#include "/usr/local/air/gpc/include/gpcurses.cons"

TYPE
#include "/usr/local/air/gpc/include/gpcurses.type"

VAR
   ekran : WINDOW;
   napis : string80;

#include "/usr/local/air/gpc/include/gpcurses.proc"

{...}

   napis := 'dowolny napis';
   WADDSTR(ekran, napis);
```

w celu skompilowania programu:

```
gpc program.p -lncurses
```


Podsumowanie: operacje terminalowe w systemie Unix

- obsługa terminali znakowych w systemie Unix: terminfo i curses
- podstawy elementy biblioteki curses: inicjalizacja, tworzenie okien, numeracja wierszy i kolumn w oknach, kursory, wyświetlanie i aktualizacja zawartości okien
- parametry i działanie podstawowych procedur: wyświetlanie, kasowanie i wczytywanie tekstów w oknach
- atrybuty: wyświetlanie tekstów z atrybutami
- klawisze funkcyjne: wczytywanie znaków z rozpoznawaniem klawiszy funkcyjnych
- tryby specjalne: `CBREAK`, `NOECHO`
- kompilacja programów z wykorzystaniem biblioteki curses