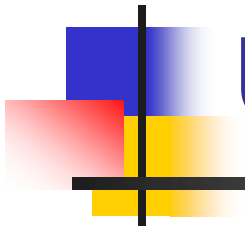


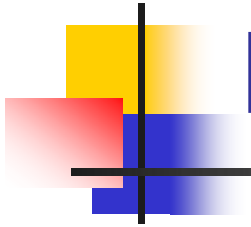
Unix-like knihovna curses

Ovládání výstupu - obrazovky

Ovládání vstupu - klávesnice

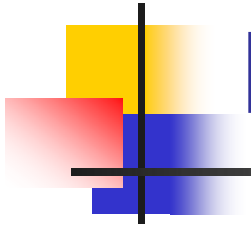
Podpůrné funkce a utility

/home/xbayer/open/11_xbayer.pdf



Knihovna curses

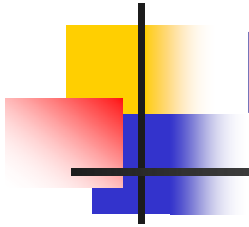
- Knihovna funkcí, struktur a konstant pro podporu „textového uživatelského rozhraní“:
  - funkce nezávislé na terminálu
  - funkce závislé na operačním systému (MS-DOS **`conio`**)
  - operace jsou v rámci možností optimalizované
  - podporuje práci s celou obrazovkou, oknem nebo “pad” (okno, které není přímo vidět)
  - umožňuje výpis, přepis textu, práci s barvami, kurzorem, ovládání vstupu (např. zastavení opisu znaku na terminál), zjištění vlastností a stavu terminálu, používání šipek či funkčních kláves, atd.
  - pokračování – **`ncurses`**



Knihovna curses (2)

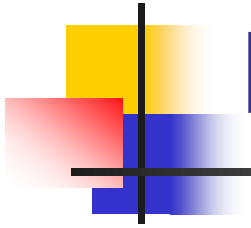
- Překládáme s „**`-lcurses`**“
  - resp. **`-lcurses_g`**, (`ncurses`: `-lncurses`, `-lncurses_g`)
- Před spuštěním na Aise nastavit: „**`export TERM=linux-aisa`**“.
- Do programu vkládáme: „**`#include <curses.h>`**“
- Celkem obsahuje přes 300 funkcí

  => zde projdeme pouze některé z nich.
- [**`ncurses`** (new curses) nahrazuje **`curses`** (vývoj přerušen)]



Knihovna curses (3)

- Knihovna dovoluje pracovat s „okny“.
  - struktura **`WINDOW`**
  - většinou používáme ukazatel na ni, tedy **`WINDOW *`**
  - jedná se o dvojdimenzionální pole znaků (viz níže)
  - „defaultní“ okno se nazývá **`stdsrc`**
    - další vytváří funkce **`newwin()`**
- Všechny znaky používané v rámci struktur WINDOW jsou typu **`chtype`**.
  - představuje vlastní znak + atributy

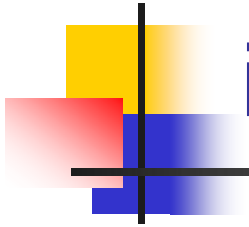


Knihovna curses - příklad

/home/xbayer/open/cervik

/home/xbayer/open/cervik.data

- http://www.mkssoftware.com/docs/man3/ncurses.3.asp
- http://web.cs.mun.ca/~rod/ncurses/ncurses.html



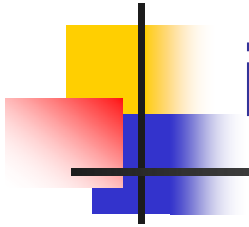
Inicializace
initscr(); cbreak(); noecho();

```
WINDOW *initscr(void);
```

- v drtivé většině první volaná funkce
  - v programu smí být volána nejvýše 1x
- rozpozná typ terminálu, inicializuje curses datové struktury, vymaže obrazovku (první refresh()), ...
- při úspěchu vrátí ukazatel na stdscr
- v případě chyby vypíše chybové hlášení a končí

```
SCREEN *newterm(char *type, FILE *out, FILE *in);
```

- používá se místo **`initscr()`** v případech:
  - kdy program vypisuje na více terminálů
  - kdy program potřebuje indikaci chybového stavu
- volá se pro každý terminál 1x



Inicializace (2)
initscr(); cbreak(); noecho();

```
int cbreak(void);
```

- zapíná „cbreak“ mód, kdy znaky zadané na vstup
  - nejsou „bufferovány“
  - jsou okamžitě k dispozici programu

```
int nocbreak(void);
```

- vypíná „cbreak“ mód, kdy znaky zadané na vstup
  - jsou „bufferovány“ do zadání konce řádku ’\n’ nebo <LF>

```
int halfdelay(int t);
```

- podobné jako „cbreak“ mód
- znak dostupný po **t/10** sekundách ([1-255])

Inicializace (3)
initscr(); cbreak(); noecho();

`void wtimeout(WINDOW *win, int t);`

- **`t < 0`** blokující, **`t == 0`** neblokující, **`t > 0`** čeká **`t`** ms

`int raw(void);`

- zapíná „raw“ mód
  - podobné jako „cbreak“ mód
  - „všechny“ znaky jsou plně propuštěny bez interpretace

`int noraw(void);`

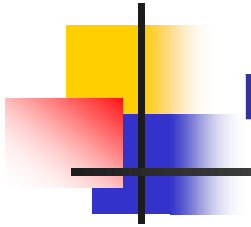
- vypíná „raw“ mód

`int noecho(void);`

- zakazuje opisování zadaného znaku funkcí **`getch()`**

`int echo(void);`

- povoluje opisování zadaného znaku funkcí **`getch()`**



Inicializace ... pokračování
nonl(); intrflush(win, 0); keypad(win, 1);

```
int nonl(void);
```

- zakazuje
  - překlad „return“ do „nového řádku“ na vstupu
  - překlad „nového řádku“ do <CRLF> na výstupu

```
int nl(void);
```

- povoluje, co **`nonl()`** zakázala

```
int intrflush(WINDOW *win, bool b);
```

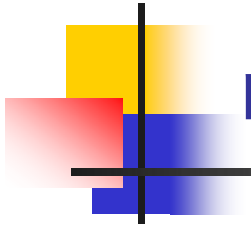
- **`b == 1`**
  - zmáčknutí „přerušovací“ klávesy způsobí „flush“ fronty znaků, rychlejší reakce na přerušení
- **`b == 0`**
  - zabránění „flush“

Inicializace ... pokračování (2)
nonl(); intrflush(win, 0); keypad(win, 1);

```
int keypad(WINDOW *win, bool b);
```

- **b == 1**
  - povoluje pro win používat funkční klávesy (šipky, ...)
  - pro specielní klávesy jsou v curses definovány symbolické konstatny (např. KEY_LEFT)
- **b == 0**
  - způsobí, že curses zpracování funkčních kláves neošetřuje
  - Interpretace zůstává klasicky na programu
- implicitní hodnota je 0

Vytvoření a zrušení okna
newwin(), delwin();



```
WINDOW *newwin(int l, int c, int y, int x);
```

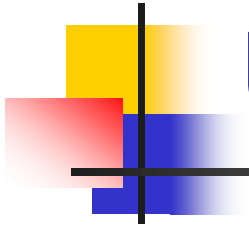
- vytvoří okno o **l** řádcích a **c** sloupcích s levým horním rohem v souřadnicích [**x**, **y**].
- „fullscreen“ lze vytvořit jako **newwin(0, 0, 0, 0);**

```
WINDOW *subwin(WINDOW *orig, int l, int c, int y, int x);
```

- vytvoří „podokno“ okna orig a vrací na něj ukazatel
- obě okna sdílí paměť, změna okna orig ovlivní obě

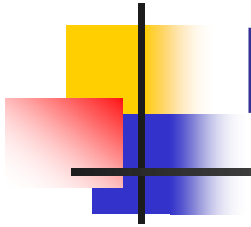
```
int delwin(WINDOW *win);
```

- uvolní alokovanou paměť spojenou s oknem **win**
- (případná „podokna“ musí být zrušena přel uvolněním okna hlavního)



Ukončení práce s curses

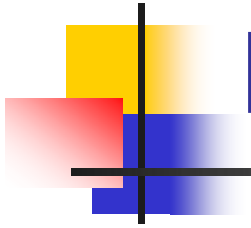
- **`int endwin(void);`**
  - program volá funkci před výstupem z curses (trvale i dočasně)
  - obnoví tty módy, posune kurzor, nastaví terminál do odpovídajícího „nevizuálního“ módu
  - (v případě užití inicializace pomocí **`newterm()`** nutno volat pro každý terminál)
- **`void delscreen(SCREEN* sp);`**
  - uvolňuje prostředky alokované pro strukturu **`SCREEN`**
  - program volá po volání **`endwin()`**



Příklad „Hello World“

/home/xbayer/open/hello_world.c

```
#include<curses.h>
...
WINDOW *w;  //nove okno
initscr();  //inicializace curses
cbreak();   //cbreak mod – nebufferujeme znaky
noecho();   //bez opisu na obrazovku
w = newwin(5, 15, 5, 5);
wborder(w,'|','|','-','-','-','-','-','-');
mvwaddstr(w,1,1,"HELLO WORLD"); //vypis textu
wrefresh(w); //zobrazime, co jsme vykreslili
napms(2000); //pockame 2000ms
endwin();    //vratime se zpet z curses
```



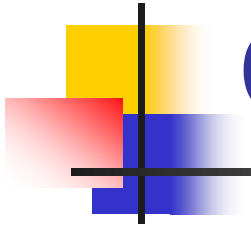
Rámeček

```
int wborder(WINDOW *win, chtype ls, chtype rs,
chtype ts, chtype bs, chtype tl, chtype tr,
chtype bl, chtype br);
```

- vykreslí kolem okna rámeček
- ls/rs – strana vlevo/strana vpravo
- ts/bs – strana nahoře/strana dole
- tl/tr – roh nahoře vlevo/vpravo
- bl/br – roh nahoře vlevo/vpravo

```
/-------------\
|HELLO WORLD  |
|             |
|             |
\-------------/
```

- `wborder(w,'|','|','-','-','/','\\','\\','/');`



Čekání, zobrazení

```
int napms(int t);
```

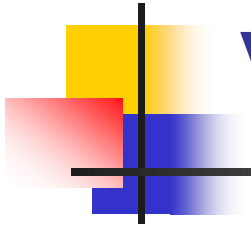
- způsobí čekání po t milisekund

```
int wrefresh(WINDOW *win);
```

- vypíše obsah 2D pole znaků na terminál
- do zavolání nejsou změny pozorovatelné

```
int refresh(void);
```

- stejné jako **`wrefresh`**
- používá implicitně **`stdsrc`** okno



Výpis textu

```
int waddch(WINDOW *win, chtype ch);
```

- vypíše znak ch na pozici kurzoru do okna win
- kurzor posune, na konci řádku odřádkuje

```
int waddstr(WINDOW *win, char *str);
```

- vypíše řetězec str do okna win
- stejné, jako výpis waddch() znak po znaku

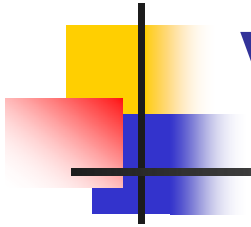
```
int mvwaddch(WINDOW *win, int y, int x, chtype c);
```

- vypíše znak c do okna win na pozici [x, y]

```
int mvwaddstr(WINDOW *w, int y, int x, char *s);
```

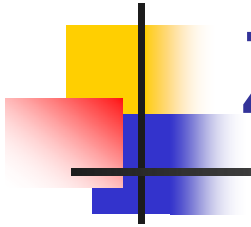
- vypíše řetězec s do okna w na pozice začínající [x, y]

/home/xbayer/open/vypis.c



Výpis textu (2)

```
int addstr(const char *str);
int addnstr(const char *str, int n);
int waddstr(WINDOW *win, const char *str);
int waddnstr(WINDOW *win, const char *str, int n);
int mvaddstr(int y, int x, const char *str);
int mvaddnstr(int y, int x, const char *s, int n);
int mvwaddstr(WINDOW *w, int y, int x, const char
  *s);
int mvwaddnstr(WINDOW *w, int y, int x, const char
  *s, int n);
```



Zakázání kurzoru, pohyb

```
int curs_set(int visibility);
```

0 – neviditelný

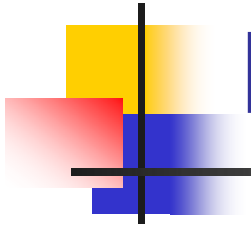
1 – normální

2 – „velmi“ viditelný

- pokud terminál neumí skrýt kurzor, vrátí ERR
- při úspěchu vrací minulý stav

```
int wmove(WINDOW *win, int y, int x);
```

- posune kurzor okna win na [x, y]
- fyzický kurzor se neposune do volání refresh()



Práce s barvami

Barvy se definují v párech:

- barva popředí
- barva pozadí

```
bool has_colors(void);
```

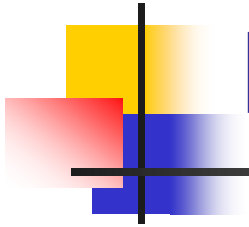
- zjistí, zda smíme použít barvy

```
int start_color(void);
```

- inicializuje práci s barvami

```
bool can_change_color(void);
```

- zjistí, zda smíme předefinovat barvy



Práce s barvami (2)

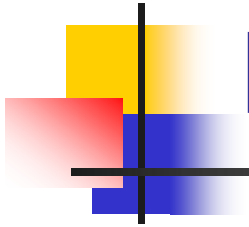
```
int init_pair(short pair, short f, short b);
```

- pair – číslo páru, které měníme [ohrainičeno **COLOR_PAIRS**]
- f – barva popředí, b – barva pozadí

```
int init_color(short c, short r, short g,
  short b);
```

- c – číslo barvy [ohrainičeno **COLOR_PAIRS**]
- r, g, b – složky RGB [0 – 1000]

- **COLOR_BLACK, COLOR_RED, COLOR_GREEN, COLOR_YELLOW, COLOR_BLUE, COLOR_MAGENTA, COLOR_CYAN, COLOR_WHITE**



Použití atributů

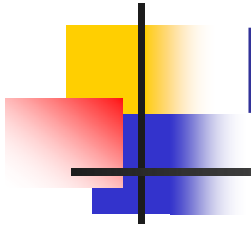
```
int attron(int attrs);
```

- zapíná pouze pojmenovaný atribut

```
int attroff(int attrs);
```

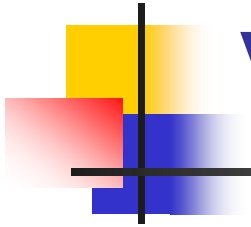
- vypíná pouze pojmenovaný atribut

- **A_STANDOUT, A_UNDERLINE, A_REVERSE, A_BLINK, A_DIM, A_BOLD, A_ALTCHARSET, A_CHARTEXT, COLOR_PAIR(n)**



Příklad barev a atributů

/home/xbayer/open/barvy.c



Vstup

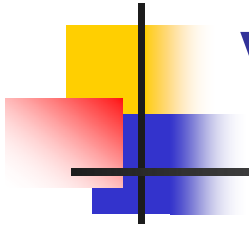
```
int wgetch(WINDOW *win);
```

- načte znak ze vstupu v okně win
- v nečekacím módu vrací ERR, pokud je vstup prázdný
- implicitně opisuje vstup, pomocí noecho() lze zakázat
- je-li keypad nastaven na 1, lze „číst“ i funkční klávesy

```
int mvwgetch(WINDOW *win, int y, int x);
```

```
int ungetch(int ch);
```

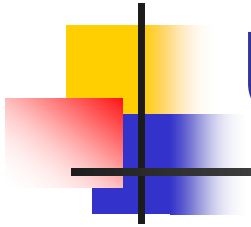
- vrátí znak zpět do bufferu



Vstup (2)

- Pro funkční klávesy jsou definovány konstanty:
- Kurzorové šipky
  - KEY_DOWN, KEY_UP, KEY_LEFT, KEY_RIGHT
- F1-F12 [1-62]
  - KEY_F(n)
- Enter
  - KEY_ENTER
- Backspace
  - KEY_BACKSPACE

man curs_getch



Úkol

- Dobrovolný úkol za mnoho kladných bodů (bude-li vypadat a fungovat dobře):
  - Naprogramujte malou počítačovou hru, můžete se inspirovat starými „kousky“ typu „space invanders“, či „FFFS“.
  - Netřeba implementovat barvy (viz potíže).
  - Netřeba implementovat OpenGL či DirectX v TUI ;-)
  - Hodnocení bude záviset na propracovanosti.
  - Vzhledem k tomu, že časový tlak je u studentů různý, je ponechána poměrně velká volnost.
  - Odevzdávat do 14ti dnů v příloze na email s malým tutoriálem, jak hru hrát.
  - Musí fungovat na Nymfách v B130, nikoli na Aise.