PRG Exam cards

Proč se programy v C rozdělují do hlavičkových souborů (.h) a zdrojových souborů (.c)? #card

Rozděluji tak deklarace a implementace. Veřejné deklarace musí být známé všem. Je to jistá specifikace komunikace mezi programy.

Jaký význam má hlavičkový soubor zdrojových souborů programu v C? #card

Rozděluji tak deklarace a implementace. Veřejné deklarace musí být známé všem. Je to jistá specifikace komunikace mezi programy. Umožňuje ostatním programům znát objekty (funkce) v knihovně bez znalosti implementace.

Jak probíhá překlad a linkování (sestavení) programu v C? #card

Kompilace vezme lidsky-čitelný textový soubor a vygeneruje objekt s instrukcemi ve strojovém kódu, který odpovídá zdrojovému kódu. Tento objekt ale není spustitelný, to zařídí linker.
Linker vezme jeden nebo více objektových souborů a vytvoří z nich spustitelný soubor. Hledá odkazy na funkce (syscally nebo z knihoven nebo napříc objekty).

Vysvětlete rozdíl mezi překladem zdrojových souborů a linkováním programu? #card

Kompilace vezme lidsky-čitelný textový soubor a vygeneruje objekt s instrukcemi ve strojovém kódu, který odpovídá zdrojovému kódu. Tento objekt ale není spustitelný, to zařídí linker.
Linker vezme jeden nebo více objektových souborů a vytvoří z nich spustitelný soubor. Hledá odkazy na funkce (syscally nebo z knihoven nebo napříc objekty).

Popište proces vytvoření spustitelného programu ze zdrojových souborů jazyka C. #card

1. Preprocessor - Expandování maker a includenutí headerů
2. Kompilace - Vytvoří assemblerový kód
3. Assembly - Přemění assemblerový kód na objekt
4. Linkování - Přetvoří objekt na spustitelný soubor. Vyřeší reference na funkce.

Jaké znáte překladače jazyka C? #card

Mingw, clang, gcc

Jak zajistíme, že se hlavičkový soubor programu v C nevloží při překladu vícekrát? #card

Pomocí preprocessorových příkazů. Kód může vypadat následovně:

```
#ifndef header_name
#define header_name
/* Code of the header*/
#endif
```

Co je to preprocesor a jaká je jeho funkce při překladu zdrojového souboru v jazyce C? #card

Preprocessor je krok procesesu vytváření spustitelného souboru ze zdrojového kódu. Stará se o příkazy začínající pomocí hashtagu například `#include` nebo `#define`. Všechny tyto příkazy popisují nějakou změnu zdrojového kódu ještě před kompilací.. Například `#define KONSTANTA 3` hledá ve zdrojovém kódu řetězec `KONSTANTA` a nahradí ho řetězcem `3`.

Jak při překladu programu kompilátorem GCC nebo Clang rozšíříme seznam prohledávaných adresářů s hlavičkovými soubory?

Pomocí kompilační vlajky `-I dir` specifikujeme další adresář, kde hledat.

Záleží u kompilace programu kompilátorem GCC nebo Clang při specifikaci adresářů s hlavičkovými soubory na jejich pořadí? #card

Záleží, adresáře z vlajek se procházejí v pořadí zleva-doprava tzn. že pokud bychom měli dva stejně pojmenované headery, byl by použit ten ve více levém adresáři

Co způsobí definování makra preprocesoru NDEBUG v souvislosti s používáním funkce assert? #card

Pokud je macro jménem NDEBUG definované při includenutí `<assert.h>`. Macro assert je vypnuté → nic nedělá. Pokud je NDEBUG nedefinované macro assert zastaví program a vypíše chybu

Jaký tvar má hlavní funkce programu v C, která se spustí při spuštění programu v prostředí s operačním systémem?

`int main(int argc, char * argv[])`

Jakou návratovou hodnotou programu v C indikujete úspěšné vykonání a ukončení programu? Proč zvolíte právě tuto hodnotu? #card

Úspěšné vykonání programu indikujeme návratovou hodnotou 0. Ostatní čísla vyjadřují dané chyby. 0 byla zvolena protože správný běh je jen jeden a možných chyb je několik.

Jak předáváme parametry programu implementovanému v jazyce C? #card

Operační systém předá argumenty volání programu z terminálu do argumentů funkce main

- `argc` - počet parametrů
- `argv` - array stringů s parametry
  Pozor, první argument je název programu (jak byl volaný z terminálu)

Existuje nějaká jiná možnost jak předat uživatelské parametry programu jinak než jako argument programu? #card

- Pomocí nastavení hodnot preprocessorových konstant při kompilaci programu
- Pomocí standartního uživatelského vstupu po spuštení programu

Jaký je rozdíl mezi staticky a dynamicky linkovaným programem implementovaným v jazyce C? #card

Statické linkování přidá dané knihovny do programu, tím zvýší jeho velkost. Dynamické linkování knihonu slinkuje až při spuštění programu.

Linkují ve výchozím nastavení překladače GCC nebo Clang statické nebo dynamické binární spustitelné soubory? #card

Defaultní chování je dynamické linkování

Jak vkládáme do zdrojového souboru programu v C hlavičkové soubory jiných modulů nebo knihoven? #card

pomocí preprocessorového macra `#include`. `#include <název>` pro systémové headery, v includes dir. `#include "název"` pro hlavičky ve stejném adresáři jako program.

Jaký rozdíl mezi použitím #include <soubor.h> a #include “soubor.h”? #card

`#include <název>` pro systémové headery, v includes dir. `#include "název"` pro hlavičky ve stejném adresáři jako program.

Popište rozdíl mezi deklarací a definicí funkce v jazyce C? #card

Deklarace specifikuje interface funkce. Jak se s ní pracuje. Definice funkci implementuje a definuje kód odpovídající funkci

Jak jsou předávány parametry funkci v jazyce C? #card

Argumenty se push-ou na stack a funkce se zavolá. Argumenty můžou být předávány:

- Hodnotou: Hodnota proměnné se zkopíruje, změna této proměnné neovlivní nic mimo proměnnou funkci
- Referencí (odkazem): V C implementované pomocí pointerů. Funkci se přidá pointer na proměnnou a tím se dá vnější proměnná měnit z vnitřku funkce.

Co je to literál a co tímto pojmem označujeme? #card

Literály jsou konstatní reprezentace proměnných v C kódu. Například:

- `int a = 20`: `20` je int literál
- `float a = 20f` je float literál
- `double a = 20.0`: `20.0` je double literál
- `char a = 'c'`: `c` je char literál.

Jak lze v jazyce C realizovat předání parametru funkci odkazem? #card

Pomocí předávání pointerem

Jak lze v jazyce C omezit viditelnost funkce pouze v rámci jednoho modulu (souboru .c)? #card

Pomocí klíčového slova `static` u deklarace funkce a její neobsažením v headeru

Je možné v jazyce C volat funkci ze sebe sama (rekurze)? #card

Ano, ale nezastavené rekurze narazí na zaplnění stacku

Při volání funkce v jazyce C jsou předávány argumenty funkce, které se stanou lokálními proměnnými. V jaké části paměti jsou tyto lokální proměnné při běhu programu uloženy? #card

Na stacku

Jaké znáte kategorie proměnných z hlediska jejich umístění v paměti? #card

Lokální proměnné → Na stacku.
Dynamicky alokované → na heap.
Statické/globální → data část paměti.

Je součástí jazyka C přímá podpora (tj. klíčové/á slovo jazyka) dynamická alokace paměti? #card

Ano, `malloc`

Jak v jazyce C dynamicky alokovat paměť za běhu programu? #card

Pomocí `malloc(size)` vrátí pointer na dynamicky alokovanou pamět velikosti `size`.

Je v jazyce C nutné uvolňovat dynamicky alokovanou paměť? Pokud ano, jak to uděláte? #card

Není to vždy třeba, protože se o to většinou postará operační systém, ale je to dobrá praxe. Můžeme to udělat pomocí `free(pointer)`

Jak zjistíme velikost reprezentace datovýchtypů v jazyce C? #card

Pomocí `sizeof(typ)` např. `sizeof(int)`

Je vždy v jazyce C velikost proměnné typu int 32 bitů? #card

Bývá to pravidlem, ale nemusí to být vždycky pravda. Je třeba kontrolovat pomocí `sizeof(int)

Kdy je velikost ukazatele v jazyce C 32-bitů a kdy 64-bitů? #card

Záleží na architektuře procesoru 32-bit nebo 64-bit

Jak funguje modifikátor static při použití v definici lokální proměnné funkce v jazyce C? #card

Lokální proměnná nebude alokovaná na stacku ale v části data. Znamená to v praxi, že ve všech voláních funkce se nebude hodnota proměnné resetovat, ale zůstane stejná.

Jak funguje modifikátor extern při definici globální proměnné v jazyce C? #card

`extern` řekne kompilátoru, že tato proměnná je z externího zdroje a slíbíme, že bude známá při linkování

Jaké znáte základní znaménkové celočíselné typy v jazyce C? #card

char, short, int, long, long long

Rozlišuje jazyk C znaménkové a neznaménkové celočíselné typy? Pokud ano, co z toho plyne? #card

Ano, rozlišuje. V praxi to znamená jiné chování při přetékání. (Zároveň taky definované chování při přetečení). A možnost dosáhnout větších kladných čísel (dvojnásobně větších). Takže pokud máme jistotu, že číslo nebude menší než 0 vyplatí se použít unsigned.

Jaké znáte neceločíselné typy v jazyce C? Jaká je jejich vnitřní reprezentace (velikost)? #card

- float: 32 bitů
- double: aspoň 64 bitů
- long double: aspoň 80 bitů

Je součástí jazyka C typ logické hodnoty ,,true/false"? Pokud ano, jak se používá? Pokud ne, jak jej definujete? #card

Existuje standartní header stdbool.h, kde jsou hodnoty true/false definované jako 1/0 (buď char nebo int v závislosti na headeru). Pracuje se s nimi tedy jako s čísly, hlavně pro if statementy, kde se dají použít pro rozhodování.

Jak v C definujete ukazatel na proměnnou, např. typu int? #card

```
int n = 10;
int * n_ptr = &n;
```

Jaké jsou v C omezení pro názvy proměnných a funkcí? #card

Můžeme používat alfanumerické znaky a podtržítka. Názvy musí začínat písmenem.

Jaké znáte escape sekvence používané v C pro řídicí znaky? #card

- `\n` : newline
- `\r` : carriage return
- `\b` : backspace
- `\t`: tab

Jak jsou v C reprezentovány textové řetězce? #card

Jako array charů zakončený null terminátorem → `\0`.

Jak v C zapisujeme identifikátory (jména funkcí a proměnných)? #card

Co to vůbec znamená?

Jakými dvěma způsoby lze v C vytvářet konstanty? #card

1. Pomocí preprocesorové direktivy `#define`
2. Pomocí klíčového slova `const`

Popište výčtový typ jazyka C. Uveďte vlastnosti, které považujete za důležité? #card

`enum dny {PO, UT, ST, CT, PA, SO, NE}`
^1653493561607
^1653493561608
přiřadí číselné hodnoty pojmenovaným konstantám. To napomáhá čitelnosti kódu. Dále se dá využít k vytvoření vlajek, které umožňují několik nastavení v jednom čísle:

```
enum options {
   A = 1,
   B = 2,
   C = 4,
   D = 8,
}


int option = 0;
option &= A; // Přidá option A a C do int option.
option &= C;
if (option & A){
 /*do stuff*/
}
```

Jaké znáte logické operátory jazyka C? Jak se zapisují? #card

- AND: `&&`
- OR: `||`
- NOT: `!`

Jaké znáte bitové operátory jazyka C? Jak se zapisují? #card

- AND: `&`
- OR: `|`
- XOR: ``
- complement: `-`
- shift left: `<<`
- shift right: `>>`

Jak v C realizujete dělení a násobení dvěma s využitím operátorů bitového posunu? #card

```c
int a = 10;
int l = a << 1; // l = 20
int r = a >> 1; // r = 5
```

Jak v C definujete složený typ (struct)? #card

```c
struct struktura {
        int a;
        char b;
};
```

Jak zavedeme nový typ struktury, např. pojmenovaný my_struct_s. #card

```c
typedef struct my_struct {
        int a;
        char b;
} my_struct_s;
```

Co je v jazyce C pointerová (ukazatelová) aritmetika a jak se používá? #card

Při přičítání odčítání čísel od pointeru můžeme posunout místo kam ukazuje v paměti o jednu velikost proměnné na kterou ukazuje. Např pokud int má velikost 4 bajty, zvýší se hodnota adresy v paměti o 8 pokud uděláme `n_ptr += 2`. Praktické využití je například při přičítání k ukazateli na začátek pole.

Jak se v C liší proměnná typu ukazatel a typu pole[] (VLA - pole variabilní délky)? #card

Deklarace `char str[] = "Ahoj PRGC"` překopíruje hodnotu literálu do str, která je uložená na stacku. Pokud bychom udělali `char * str = "Ahoj PRGC"` neměli bychom upravovat hodnotu, protože ukazatel ukazuje na string literál někde v paměti, která nemusí být writeable. Zároveň můžeme měnit velikost `char str[]` (VLA), narozdíl od pointeru, kde bychom museli využít dynamickou paměť.

Jak v C přistupujeme k datovým položkám složeného typu (struct)? #card

Pomocí `.` například `my_struct.polozka`.

Uveďte příklad přístupu k položkám proměnné složeného typu (struct) a proměnné typu ukazatel na složený typ. #card

```
// Typ je složený typ
int polozka = my_struct.polozka;
// Typ je ukazatel na slozeny typ
int polozka = my_struct->polozka;
```

Jaký v C rozdíl mezi typy struct a union? #card

Union slouží k deklaraci proměnné která může mít různý typ, například `int` nebo `char`. Union by poté měla velikost intu (toho největšího typu). Struct ukládá několik typů za sebou. Například `int` a `char`. Má poté velikost `sizeof(int) + sizeof(char)`

Stručně popište typ union používaný v jazyce C. #card

Union slouží k deklaraci proměnné která může mít různý typ, například `int` nebo `char`. Union by poté měla velikost intu (toho největšího typu).

```
union data {
   int i;
   char c;
}
^1653493561629


sizeof(union data) // equals to sizeof(int)
```

Jak se v jazyce C používá operátor přetypování? #card

```
int n = 20;
int n_ptr = &n;
char * n_bytes = (char *) n_ptr; // přemění int na array jeho bajtů
```

Co v C reprezentuje typ void? #card

Reprezentuje žádný typ. Co to prakticky znamená záleží na kontextu:

- `void foo(int a)` - funkce co nic nevrací
- `int foo(void)` - funkce co vrací int, ale nemá žádný argument

Co v C reprezentuje typ void*? #card

Reprezentuje ukazatel na místo v paměti, které má neznámý typ. Nejčastěji se s ním asi setkáme jako s návratovou hodnotou malloc()

Jak v C realizujete opuštění dvou nebo více vnořených cyklů z nejvnitřnějšího cyklu? #card

Existují dva způsoby:

1. `return` lze použít uvnitř funkce, pokud z ní chceme pryč
2. `goto label` skočí z místa na místo v kódu

Co v C reprezentuje identifikátor NULL? #card

Reprezentuje pointer s adresou 0. To se může hodit například u dynamicky alokované paměti, kde `free(NULL)` nedělá nic.

Jak nastavíte proměnnou typu ukazatel na prokazatelně neplatnou hodnotu? #card

Tak, že ji nastavíme na `NULL`

Napište základní tvar hlavní funkce main, která se používá v C programech? Uveďte další možné tvary. #card

```
int main(int argc, char* argv[]){}
int main(void){}
void main(void){}
void main(int argc, char* argv[]){}
```

Jaký je rozdíl mezi ukazatelem na konstantní hodnotu a konstantním ukazatelem? #card

Ukazatel na konstatní hodnotu může měnit kam ukazuje, ale ta hodnota na kterou ukazuje by měla být konstatní. Konstantní ukazatel na hodnotu ukazuje pořád stejně, ale hodnota na kterou ukazuje se může měnit

Jak v C zapíšete konstantní ukazatel na konstantní hodnotu, např., typu double? #card

```
const int n = 20;
const int* const ptr = &n
```

Co je v C ukazatel na funkci? K čemu slouží a jak definujete proměnou typu ukazatel na funkci? #card

```
void fun(int a){
   /*do sth*/
}
^1653495873260

//definuje pointer názvu fun_ptr na funkci, která bere za argument int
// a returnuje void
void (*fun_ptr)(int);
fun_ptr = &fun;
(*fun_ptr)(10); /* volá funkci na pointeru s argumentem 10 */
```

Nejčastěji se s ní setkáme asi u vytváření vláken kterým se předává jejich funkce, co mají vykonávat. Nebo například řazení arraye, kde každou hodnotu, dle které řadíme, získáme pomocí zavolání funkce na ten prvek.

Je v C rozdíl definovat složený typ pouze jako struct a prostřednictvím typedef struct? Pokud ano, tak jaký? #card

Pokud definujeme pomocí typedef struct nemusíme při dalším používáním typu psát `struct název`, ale stačí `název`

Můžeme v C při definici proměnné typu pole, proměnnou přímo inicializovat? Pokud ano, jak? #card

Ano, můžeme

```
int pole[] = {1,2,3,4};
```

Můžeme v C při definici proměnné typu struct inicializovat pouze určitou položku? #card

Ano, můžeme

Jakou funkcí v C vytisknete na obrazovku formátovaný znakový výstup? V jaké standardní knihovně je funkce definována? #card

Funkcí `printf`, z knihovny `stdio`

Jak v C načtete hodnotu textového řetězce a celého čísla od uživatele? #card

```
int n = 0;
char str[256];
scanf("%s %i", str, &n)
```

Jak v C vytisknete textový řetězec na standardních výstup a standardní chybový výstup? Jakou funkci k tomu použijete? #card

```
fprintf(stderr, "Zpráva\n");
```

V jakém kontextu se používá klíčové slovo break? #card

`break` se používá v kontextu while a for loopu na jeho predčasné ukončení

V jakém kontextu se používá klíčové slovo case? #card

`case` se používá v kontextu `switch` pro definici jednotlivých případů

V jakém kontextu se používá klíčové slovo continue? #card

`continue` se používá v kontextu `while` a `for` loopu k přeskočení na další iteraci

V jakém kontextu se používá slovo default #card?

`default` se používá v kontextu `switch` jako defaultní případ, pokud žádný `case` nesedí.

V jakém kontextu se používá klíčové slovo do? #card

V kontextu `do while` loopu.

V jakém kontextu se používá klíčové slovo while? #card

Pro vytvoření `while` loopu.

V jakém kontextu se používá klíčové slovo for? #card

Pro vytvoření `for` loopu.

Jaký význam má uvedení specifikátoru register? #card

^1653495873285
Můžeme doporučit kompilátoru aby načetl hodnotu do registru a tím s ní pracoval rychleji. Nemusí nás ale poslechnout

Kdy lze použít příkaz skoku goto? #card

Ke skoku mezí částmi kódu uvnitř jedné funkce.

Co vrací operátor sizeof? #card

`sizeof(type)` vrací objekt typu `size_t`, číslo které reprezentuje počet bajtů typu.

Lze použít proměnnou jako argument operátoru sizeof? #card

Lze

Jak můžeme zjistit konkrétní velikost určitého datového typu? Např. celočíselný int nebo short. #card

Pomocí operátoru `sizeof`

Jak zajistíme, že lokální proměnná ve funkci si zachová hodnotu i při opuštění funkce? #card

Pomocí použití klíčového slova `static`

Co reprezentuje klíčové slovo void? #card

Reprezentuje žádný typ. Co to prakticky znamená záleží na kontextu:

- `void foo(int a)` - funkce co nic nevrací
- `int foo(void)` - funkce co vrací int, ale nemá žádný argument

Jak definujete konstantní hodnotu typu float? #card

`const float pi = 3.14`

Jak rozlišíte literál typu float a double? #card

```
0.123f // float
0.123 // double
```

Jak vytisknete hodnotu ukazatele na standardní výstup? Jaký formátovací příkaz v printf použijete? #card

```
int n = 20;
int *p = &n;
printf("%x\n", p);
```

Jak vytisknete hodnotu proměnné typu int, na kterou odkazuje ukazatel? #card

```
int n = 20;
int *p = 20;
printf("%i", *p);
```

Jak získáte ukazatel na proměnnou definovanou jako double d = 12.3; #card

```
double * d_ptr = &d;
```

Jak přistoupit na položku number proměnné data typu struktura? #card

```
data.number;
```

K čemu slouží modifikátor const? #card

`const` naznačuje že by se proměnná neměla měnit. A znemožňuje její změnu tradičnímy způsoby. Dá se ale obejít

Jak se v C předává pole funkcím? #card

Pomocí pointeru na první prvek pole

```
int arr[] = {1,2,3}
^1653780185953

void foo(int * pole){
        //...
}
foo(arr);
```

Je velikost paměťové reprezentace typu struct vždy součet velikostí typů jednotlivých položek? #card

Nemusí tomu tak být, kompilátor může mírně zvětšovat velikost structu tak aby lépe seděla do paměti na kulaté násobky a práce s ní byla rychlejší. tento "padding" se dá vypnout

Podporuje jazyk C přetěžování jmen funkcí? Pokud ano, od jaké verze? #card

Nepodporuje, dá se ale obejít pomocí defineů a `_Generic`

Jak probíhá proces spuštění programu implementovaného v jazyce C? #card

Popište jak v C probíhá volání funkce int doit(int r)? Jaká data jsou předávána do/z funkce a kam jsou hodnoty ukládány?

1. Na stack se zkopíruje hodnota int r
2. začne vykonávání funkce.
3. po skončení funkce se returnutá hodnota dá zkopíruje zpátky na havní stack do proměnné, kam se ukládá.

Vysvětlete rozdíl mezi proměnnou a ukazatelem na proměnnou v jazyce C? #card

Proměnná je datový typ, který drží informace o datech, dané proměnné. Například `char c = 'd'` je 8 bitů paměti, které reprezentují znak `'c'`. Ukazatel ukazuje na místo v paměti, kde můžeme data o proměnné najít. Má 32/64 bitů a popisuje adresu.

Jak v C dynamicky alokujete paměť pro uložení posloupnosti 20 hodnot typu data_t ? Jak následně takové dynamické

pole zvětšíte pro uložení dalších 10 položek? #card

```
data_t * data_ptr = malloc(20 * sizeof(data_t));
data_t * tmp = realloc(data_ptr, 30 * sizeof(data_t));
if (tmp != NULL){
        data_ptr = tmp;
}
free(data_ptr);
```

Vyjmenujte základní paměťové třídy, ve kterých mohou být uloženy hodnoty proměnných. #card

auto, static, extern, register

Jaký význam má klíčové slovo static v závislosti na kontextu? #card

Statická glob. proměnná a funkce jsou pouze viditelné v jednom daném souboru.
Statická lokální proměnná je persistentní napříč voláními funkce

Definujte pole variabilní délky o velikosti n, kterou načtete ze standarního vstupu. #card

```
int n = 0;
scanf("%i", &n);
int arr[n];
```

Definujte diagonální (jednotkovou) matici 3×3 jako 2D pole typu int. #card

```
Definujte diagonální (jednotkovou) matici 3×3 jako 2D pole typu int.
```

Garantuje uvedení const u definice proměnné, že není žádná možnost jak příslušnou hodnotu proměnné změnit? #card

Negarantuje

Je v C možné použít příkaz nepodmíněného skoku goto ke skoku z jedné funkce do jiné? Pokud ne, proč? #card

Je to nedefinované chování. Nemuselo by být vůbec jasné jaké hodnoty pro proměnné v té funkci používat. Například při rekurzi

```
int i=0;
A(){
run:
    B();
}
B(){
if(i==10)
    goto run;
i++;
A();
}
```

Do jaké verze A by mělo toto goto skočit? Není vůbec jasné

Popište k čemu slouží příkaz dlouhého skoku (longjmp/setjmp) v C. Jak se používá? #card

`setjmp` nastaví místo, kam má skákat `longjmp`. setjmp uloží současný kontext a stav a `longjmp` na něj skočí a obnoví.

Vyjmenujte základní rozdělení paměti přidělné spuštěnému programu z hlediska kódu, proměnných a literálů? #card

- Text - instrukce programu
- Inicializovaná data - globální proměnné, statické proměnné, literály
- Neinicializovaná data - bss - jsou zde uložené neinicializované proměnné
- Stack - Ukládá lokální proměnné.
- Heap/halda - Dynamicky alokovaná paměť

Vyjmenujte (čtyři) specifikátory paměťové třídy (Storage Class Specifiers - SCS). #card

- auto
- register
- static
- extern

Jaké typy paměti dle způsobu alokace rozlišujeme v jazyce C? #card

Haldu a stack. Na stacku jsou lokální proměnné na heap jsou dynamicky alokovaná data

Definujte nový typ, který umožní sdílet paměť pro proměnnou typu double, nebo proměnnou typu int. #card

```
typedef union {
        dbl double;
        intgr int;
} doublint;
```

Co znamená klíčové slovo volatile? #card

Říká kompilátoru, že proměnná se může kdykoliv změnit a musí s tím počítat.

Jaký význam má klíčové slovo extern dle kontextu? #card

1. U proměnné jí deklaruje, ale nedefinuje. Řekne compileru, že je v jiném souboru a bude linknutá později
2. U funkcí umožňuje aby byly volané z jiných souborů, defaultní chování

V jakém hlavičkovém souboru standardní knihovny C jsou deklarovány funkce pro vstup a výstup? #card

`stdio.h`

V jakém hlavičkovém souboru standardní knihovny C jsou deklarovány nejběžnější funkce std. knihovny? #card

`stdlib.h`

V jakém hlavičkovém souboru standardní knihovny C jsou deklarovány funkce pro práci s textovými řetězci? #card

`string.h`

Co je errno a v jakém hlavičkovém souboru standardní knihovny C je deklarováno? #card

`errno.h` je to číslo, které popisuje poslední chybu například ze systémových knihoven.

Jakým způsobem jsou předávány nebo jinak ukládány chybové stavy ve většině funkcí standardní knihovny C? #card

Jakým způsobem jsou předávány nebo jinak ukládány chybové stavy ve většině funkcí standardní knihovny C? #card

Buďto použitím návratové hodnoty nebo pomocí `errno`

K čemu slouží makro assert a v jakém je hlavičkovém souboru standardní knihovny C? #card

`assert.h`. Zajišťuje pravdivost nějaké podmínky, užitečné při debugování. Například pro kontrolu otevření sériového portu

Ve kterém hlavičkovém souboru standardní knihovny C jsou definovány matematické funkce? #card

`math.h`

Ve kterém hlavičkovém souboru standardní knihovny C byste hledali rozsahy základní číselných typů? #card

`limits.h`

Jakým způsobem otevřete soubor pro čtení? Napište krátký (1-3 řádkový) kód? #card

```
FILE *f = fopen("filename", "r");
```

Jakým způsobem otevřete soubor pro zápis? Napište krátký (1-3 řádkový) kód? #card

```
FILE *f = fopen("filename", "w");
```

Proč je vhodné explicitně zavírat otevřený soubor? Jakou funkci standardní knihovny C k tomu použijete? #card

Použiju funkci `fclose`. Můžeme ztratit data, které se nezapíšou do souboru nebo nám můžou dojít file descriptory.

Jak zjistíte, že jste při čtení souboru dosáhli konce souborů? Jakou funkci standardní knihovny C k tomu můžete použít? #card

Pokud získáme při čtění ze souboru symbol rovný `EOF` soubor končí. Funkce `feof` zkontroluje jestli jsme opravdu na konci souboru

Jak zjistíte podrobnosti o selhání čtení/zápisu z/do souboru s využitím funkcí standardní knihovny C? #card

Pomocí funkce `ferror`, která dá číslo reprezentující chybový stav

Jak rozlišíte chybu a dosažení konce souboru při neúspěchu čtení ze souboru, např. funkcí fscanf()? #card

Při chybě a konci fscanf vrátí `EOF`, můžeme pak pomocí `ferror` zjistit, zda-li to byla chyba nebo konec

Jaké znáte funkci/e standardní knihovny C pro náhodný přístup k souborům? #card

??? `fseek?`

Jaké znáte funkce standardní knihovny C pro blokové čtení a zápis? #card

`read` a `write`

Co je to proces v terminologii operačního systému? #card

Program, který právě běží a má svůj prostor v paměti

Budete se snažit svůj program paralelizovat i když máte pouze jeden procesor? Svou odpověď zdůvodněte. #card

???

Jaké základní operace související s paralelním programováním (více procesové/vláknové) řeší

programovací jazyky s explicitní podporou paralelismu? #card

Podpora vytváření nových procesů, zastavování dětí, když rodič zemře, určení sdílení paměti mezi parentem a dítětem

Jaké entity (operačního systému) slouží k řízení přístupu ke sdíleným zdrojům? #card

Semafor,

Jak lze standardní vstup a výstup využít pro komunikaci mezi procesy? #card

Pomocí pipeování na linuxových systémech

```
./program1 | ./program2
```

Co je to vlákno (thread)? #card

Nezávislé vykonávání instrukcí části programu, které může managovat scheduler

Jaký je rozdíl mezi vláknem a procesem? #card

Proces je:

1. heavyweight
2. seperátní paměť a zdroje
3. `fork()`
   Vlákno:
4. ligtweight
5. sdílená paměť
6. `clone()`

Co musí úloha splňovat, aby mělo smysl uvažovat o vícevláknové architektuře aplikace (obecně, ne konkrétní typ aplikací)? #card

Vyskytuje se v ní čekání na uživatelský vstup/jiná data a aplikace musí být responsivní i při čekání na tyto data

Je aplikace s interaktivním rozhraním vhodným kandidátem pro vícevláknovou aplikaci a proč? #card

Ano, aplikace vyžaduje aby byla stále responsivní na uživatelský vstup, i když zrovna něco dělá.

Kdy nemá smysl použití více vláken pro aplikaci s uživatelským rozhraním? #card

Pokud akce, co uživatel vykonává nezpusobí žádný dlouhý výpočet a aplikace bude znovu brzo responsivní.

Má smysl vyvíjet vícevláknové aplikace pro systémy s jediným CPU a proč? #card

Ano, i jedno CPU může čekat na data a nebo na webový request. Nebo vyžadovat okamžitou reakci na uživatelský vstup.

Kde se mohou nacházet vlákna z hlediska řízení přidělování procesoru? #card

V userspace (user knihovna) nebo implementovaný operačním sys (scheduler).

Jak jsou rozvrhována vlákna řešená uživatelskou knihovnou a co to znamená z hlediska priority vláken? #card

Knihovna si rozvrhování řeší vlastním rozvrhovačem. Proces musí utrácet čas na rozhodování priority průběhu vláken.

Jaké modely vícevláknových aplikací znáte? #card

- Boss/Worker - Jedno hlavní vlákno řídí ostatní
- Peer - Vlákna si rozvrhují práci bez řídícího
- Pipeline - spracovávání dát v sekvenci operací

Co je to thread pool a k čemu je dobrý #card

Předvytvořená zásoba vláken, které je postupně přiřzovaná práce. Snižuje overhead vytváření nových vláken při každém požadavku

Jak snížíte nároky opakovaného vytváření vláken? #card

Pomocí thread poolu. tj. Předvytvořená zásoba vláken, které je postupně přiřzovaná práce. Snižuje overhead vytváření nových vláken při každém požadavku

Jakou architekturu vícevláknové aplikace použijete v případě zpracování proudu dat? #card

pipeline

Jaké vlastnosti musí splňovat proudové zpracování dat, aby bylo výhodné použít více vláken? #card

Data musí být rozdělitelné na části které jsou na sebe nezávislé při zpracovávní.

Jak předáváme data mezi vlákny v úloze producent/konzument? #card

Pomocí nějakého bufferu, kam se postupně připisuje práce. Buffer refrencí na datové jednotky

Jaké je základní primitivum synchronizace více vláken? #card

Semafor

Jaké znáte primitiva pro synchronizaci více vláken? #card

Mutex, cond variable, semafor

Kdy říkáme, že je funkce reentrantní? #card

V jedné chvíli ta funkce může být vykonávána několikrát

Co je to thread-safe funkce? #card

Funkce, kterou může volat několik vláken najednou

Jak dosáhneme reentrantní funkce? #card

Nesmíme psát do statických a nesmíme používat globální data

Jak dosáhneme thread-safe funkce? #card

Musíme používat synchronizační primitivy při přístupu ke globálním datům

Jaké hlavní synchronizační problémy se objevují u vícevláknových aplikací? #card

- dead lock - vlákno čeká na mutex1, který je už zamklý druhým vláknem. Aby ho odemklo, musí první vlákno odemknout jiný mutex. Ale to už je seklé na mutex1
- race condition - Přístup několika vláken najednou do sdílené paměti

Co je to problém uváznutí (deadlock)? #card

- dead lock - vlákno čeká na mutex1, který je už zamklý druhým vláknem. Aby ho odemklo, musí první vlákno odemknout jiný mutex. Ale to už je seklé na mutex1

Co je to problém souběhu (race conditions) u vícevláknové aplikace? #card

- race condition - Přístup několika vláken najednou do sdílené paměti

Jak se lze vyhnout problému uváznutí u vícevláknové aplikace? #card

Používat co nejkratší kritické sekce → zamykat a odemykat co nejblíže kritickému kódu.