

U. Domácí úloha

Obsah

1. [Anotace](#)

2. [Zadání](#)

[Detailní specifikace](#)

1. [Autentizace](#)
2. [Pohyb robota k cíli](#)
3. [Vyzvednutí tajného vzkazu](#)
4. [Dobíjení](#)

[Chybové situace](#)

1. [Chyby při autentizaci](#)
2. [Syntaktická chyba](#)
3. [Logická chyba](#)
4. [Timeout](#)

5. [Speciální situace](#)

6. [Optimalizace serveru](#)

7. [Ukázka komunikace](#)

[Testování](#)

1. [Tester](#)
2. [Ke stažení](#)

9. [Požadavky na řešení](#)

[Odevzdání](#)

1. [Nahrání na archivační server](#)
2. [Osobní prezentace](#)

Anotace

Cílem úlohy je vytvořit vícevláknový server pro TCP/IP komunikaci a implementovat komunikační protokol podle dané specifikace. Pozor, implementace klientské části není součástí úlohy! Klientskou část realizuje testovací prostředí.



Server nemusí být opravdu vícevláknový, musí jen zvládat obsluhovat několik klientů najednou. Jestli toho dosáhnete v jednom vlákně nebo za pomoci procesů je úplně jedno, hlavně když projdete všemi testy.



Před začátkem implementace si prostudujte [poznámky k odevzdání](#)! Ušetříte si budoucí komplikace.



Informace o tom jak psát client-server komunikaci naleznete na [stránce s prosemináři](#) ([./seminars/index.html#_ukázková-implementace-v-c](#)).

Zadání

Vytvořte server pro automatické řízení vzdálených robotů. Roboti se sami přihlašují k serveru a ten je navádí ke středu souřadnicového systému. Pro účely testování každý robot startuje na náhodných souřadnicích a snaží se dojít na souřadnici [0,0]. Na cílové souřadnici musí robot vyzvednout tajemství. Po cestě k cíli mohou roboti narazit na překážky, které musí obejít. Server zvládne navigovat více robotů najednou a implementuje bezchybně komunikační protokol.

Detailní specifikace

Komunikace mezi serverem a roboty je realizována plně textovým protokolem. Každý příkaz je zakončen dvojicí speciálních symbolů „\a\b“. (Jsou to **dva** znaky 'a' a 'b'.) Server musí dodržet komunikační protokol do detailu přesně, ale musí počítat s nedokonalými firmwary robotů (viz sekce Speciální situace).

Zprávy serveru:

Název	Zpráva	Popis
SERVER_CONFIRMATION	<16-bitové číslo v decimální notaci>\a\b	Zpráva s potvrzovacím kódem. Může obsahovat maximálně 5 čísel a ukončovací sekvenci \a\b.
SERVER_MOVE	102 MOVE\a\b	Příkaz pro pohyb o jedno pole vpřed
SERVER_TURN_LEFT	103 TURN LEFT\a\b	Příkaz pro otočení doleva
SERVER_TURN_RIGHT	104 TURN RIGHT\a\b	Příkaz pro otočení doprava
SERVER_PICK_UP	105 GET MESSAGE\a\b	Příkaz pro vyzvednutí zprávy
SERVER_LOGOUT	106 LOGOUT\a\b	Příkaz pro ukončení spojení po úspěšném vyzvednutí zprávy
SERVER_KEY_REQUEST	107 KEY REQUEST\a\b	Žádost serveru o Key ID pro komunikaci
SERVER_OK	200 OK\a\b	Kladné potvrzení
SERVER_LOGIN_FAILED	300 LOGIN FAILED\a\b	Nezdařená autentizace
SERVER_SYNTAX_ERROR	301 SYNTAX ERROR\a\b	Chybná syntaxe zprávy
SERVER_LOGIC_ERROR	302 LOGIC ERROR\a\b	Zpráva odeslaná ve špatné situaci

SERVER_KEY_OUT_OF_RANGE_ERROR 303 KEY OUT OF RANGE\ab Key ID není v očekávaném rozsahu

Zprávy klienta:

Název	Zpráva	Popis	Ukázka	Maximální délka
CLIENT_USERNAME	<user name>\ab	Zpráva s uživatelským jménem. Jméno může být libovolná sekvence znaků kromě dvojice \ab a nikdy nebude shodné s obsahem zprávy CLIENT_RECHARGING.	Umpa_Lumpa\ab	20
CLIENT_KEY_ID	<Key ID>\ab	Zpráva obsahující Key ID. Může obsahovat pouze celé číslo o maximálně třech cifrách.	2\ab	5
CLIENT_CONFIRMATION	<16-bitové číslo v decimální notaci>\ab	Zpráva s potvrzovacím kódem. Může obsahovat maximálně 5 čísel a ukončovací sekvenci \ab.	1009\ab	7
CLIENT_OK	OK <x> <y>\ab	Potvrzení o provedení pohybu, kde x a y jsou celočíselné souřadnice robotu po provedení pohybového příkazu.	OK -3 -1\ab	12
CLIENT_RECHARGING	RECHARGING\ab	Robot se začal dobíjet a přestal reagovat na zprávy.		12
CLIENT_FULL_POWER	FULL POWER\ab	Robot doplnil energii a opět přijímá příkazy.		12
CLIENT_MESSAGE	<text>\ab	Text vyzvednutého tajného vzkazu. Může obsahovat jakékoliv znaky kromě ukončovací sekvence \ab a nikdy nebude shodné s obsahem zprávy CLIENT_RECHARGING.	Haf!\ab	100

Časové konstanty:

Název	Hodnota [s]	Popis
TIMEOUT	1	Server i klient očekávají od protistrany odpověď po dobu tohoto intervalu.
TIMEOUT_RECHARGING	5	Časový interval, během kterého musí robot dokončit dobíjení.

Komunikaci s roboty lze rozdělit do několika fází:

Autentizace

Server i klient oba znají pět dvojic autentizačních klíčů (nejedná se o veřejný a soukromý klíč):

Key ID	Server Key	Client Key
0	23019	32037
1	32037	29295
2	18789	13603
3	16443	29533
4	18189	21952

Každý robot začne komunikaci odesláním svého uživatelského jména (zpráva CLIENT_USERNAME). Uživatelské jméno může být libovolná sekvence 18 znaků neobsahující sekvenci „\a\b“. V dalším kroku vyzve server klienta k odeslání Key ID (zpráva SERVER_KEY_REQUEST), což je vlastně identifikátor dvojice klíčů, které chce použít pro autentizaci. Klient odpoví zprávou CLIENT_KEY_ID, ve které odešle Key ID. Po té server zná správnou dvojici klíčů, takže může spočítat "hash" kód z uživatelského jména podle následujícího vzorce:

Uživatelské jméno: Mnau!

ASCII reprezentace: 77 110 97 117 33

Výsledný hash: $((77 + 110 + 97 + 117 + 33) * 1000) \% 65536 = 40784$

Výsledný hash je 16-bitové číslo v decimální podobě. Server poté k hashi přičte klíč serveru tak, že pokud dojde k překročení kapacity 16-bitů, hodnota jednoduše přeteče (následuje ukázka pro Key ID 0):

$(40784 + 23019) \% 65536 = 63803$

Výsledný potvrzovací kód serveru se jako text pošle klientovi ve zprávě SERVER_CONFIRM. Klient z obdrženého kódu vypočítá zpátky hash a porovná ho s očekávaným hashem, který si sám spočítal z uživatelského jména.

Pokud se shodují, vytvoří potvrzovací kód klienta. Výpočet potvrzovacího kódu klienta je obdobný jako u serveru, jen se použije klíč klienta (následuje ukázka pro Key ID 0):

$(40784 + 32037) \% 65536 = 7285$

Potvrzovací kód klienta se odešle serveru ve zprávě CLIENT_CONFIRMATION, který z něj vypočítá zpátky hash a porovná jej s původním hashem uživatelského jména. Pokud se obě hodnoty shodují, odešle zprávu SERVER_OK, v opačném případě reaguje zprávou SERVER_LOGIN_FAILED a ukončí spojení. Celá sekvence je na následujícím obrázku:

Klient	Server

CLIENT_USERNAME	--->
	<--- SERVER_KEY_REQUEST
CLIENT_KEY_ID	--->
	<--- SERVER_CONFIRMATION
CLIENT_CONFIRMATION	--->
	<--- SERVER_OK
	nebo

SERVER_LOGIN_FAILED

.
.
.

Server dopředu nezná uživatelská jména. Roboti si proto mohou zvolit jakékoliv jméno, ale musí znát sadu klíčů klienta i serveru. Dvojice klíčů zajistí oboustranou autentizaci a zároveň zabrání, aby byl autentizační proces kompromitován prostým odposlechem komunikace.

Pohyb robota k cíli

Robot se může pohybovat pouze rovně (SERVER_MOVE) a je schopen provést otočení na místě doprava (SERVER_TURN_RIGHT) i doleva (SERVER_TURN_LEFT). Po každém příkazu k pohybu odešle potvrzení (CLIENT_OK), jehož součástí je i aktuální souřadnice. Pozice robota není serveru na začátku komunikace známa. Server musí zjistit polohu robota (pozici a směr) pouze z jeho odpovědí. Z důvodů prevence proti nekonečnému bloudění robota v prostoru, má každý robot omezený počet pohybů (pouze posunutí vpřed). Počet pohybů by měl být dostatečný pro rozumný přesun robota k cíli. Následuje ukázka komunikace. Server nejdříve pohne dvakrát robotem kupředu, aby detekoval jeho aktuální stav a po té jej navádí směrem k cílové souřadnici [0,0].

Klient		Server

		.
		.
		.
	<---	SERVER_MOVE
		nebo
		SERVER_TURN_LEFT
		nebo
		SERVER_TURN_RIGHT
CLIENT_OK	---	>
	<---	SERVER_MOVE
		nebo
		SERVER_TURN_LEFT
		nebo
		SERVER_TURN_RIGHT
CLIENT_OK	---	>
	<---	SERVER_MOVE
		nebo
		SERVER_TURN_LEFT
		nebo
		SERVER_TURN_RIGHT
		.
		.
		.

Těsně po autentizaci robot očekává alespoň jeden pohybový příkaz - SERVER_MOVE, SERVER_TURN_LEFT nebo SERVER_TURN_RIGHT! Nelze rovnou zkoušet vyzvednout tajemství. Po cestě k cíli se nachází mnoho překážek, které musí roboti překonat objížděkou. Pro překážky platí následující pravidla:

- Překážka okupuje vždy jedinou souřadnici.
- Je zaručeno, že každá překážka má všech osm okolních souřadnic volných (tedy vždy lze jednoduše objet).
- Je zaručeno, že překážka nikdy neokupuje souřadnici [0,0].
- Pokud robot narazí do překážky více než dvacetkrát, poškodí se a ukončí spojení.

Překážka je detekována tak, že robot dostane pokyn pro pohyb vpřed (SERVER_MOVE), ale nedojde ke změně souřadnic (zpráva CLIENT_OK obsahuje stejné souřadnice jako v předchozím kroku). Pokud se pohyb neprovede, nedojde k odečtení z počtu zbývajících kroků robota.

Vyzvednutí tajného vzkazu

Poté, co robot dosáhne cílové souřadnice [0,0], tak se pokusí vyzvednout tajný vzkaz (zpráva SERVER_PICK_UP). Pokud je robot požádán o vyzvednutí vzkazu a nenachází se na cílové souřadnici, spustí se autodestrukce robota a komunikace se serverem je přerušena. Při pokusu o vyzvednutí na cílové souřadnici reaguje robot zprávou CLIENT_MESSAGE. Server musí na tuto zprávu zareagovat zprávou SERVER_LOGOUT. (Je zaručeno, že tajný vzkaz se nikdy neshoduje se zprávou CLIENT_RECHARGING, pokud je tato zpráva serverem obdržena po žádosti o vyzvednutí jedná se vždy o dobíjení.) Poté klient i server ukončí spojení. Ukázka komunikace s vyzvednutím vzkazu:

```
Klient                Server
-----
                .
                .
                .
        <---  SERVER_PICK_UP
CLIENT_MESSAGE  --->
        <---  SERVER_LOGOUT
```

Dobíjení

Každý z robotů má omezený zdroj energie. Pokud mu začne docházet baterie, oznámí to serveru a poté se začne sám ze solárního panelu dobíjet. Během dobíjení nereaguje na žádné zprávy. Až skončí, informuje server a pokračuje v činnosti tam, kde přestal před dobíjením. Pokud robot neukončí dobíjení do časového intervalu TIMEOUT_RECHARGING, server ukončí spojení.

```
Klient                Server
-----
CLIENT_USERNAME  --->
                <---  SERVER_CONFIRMATION
CLIENT_RECHARGING --->
```

...

```
CLIENT_FULL_POWER --->
CLIENT_OK            --->
                      <---  SERVER_OK
                          nebo
                          SERVER_LOGIN_FAILED
                      .
                      .
                      .
```

Další ukázka:

```

Klient                Server
-----
                      .
                      .
                      .
                      <---  SERVER_MOVE
CLIENT_OK             --->
CLIENT_RECHARGING     --->
...

CLIENT_FULL_POWER --->
                      <---  SERVER_MOVE
CLIENT_OK            --->
                      .
                      .
                      .
```

Chybové situace

Někteří roboti mohou mít poškozený firmware a tak mohou komunikovat špatně. Server by měl toto nevhodné chování detekovat a správně zareagovat.

Chyby při autentizaci

Pokud je ve zprávě CLIENT_KEY_ID Key ID, který je mimo očekávaný rozsah (tedy číslo, které není mezi 0-4), tak na to server reaguje chybovou zprávou SERVER_KEY_OUT_OF_RANGE_ERROR a ukončí spojení. Za číslo se pro zjednodušení považují i záporné hodnoty. Pokud ve zprávě CLIENT_KEY_ID není číslo (např. písmena), tak na to server reaguje chybou SERVER_SYNTAX_ERROR.

Pokud je ve zprávě CLIENT_CONFIRMATION číselná hodnota (i záporné číslo), která neodpovídá očekávanému potvrzení, tak server pošle zprávu SERVER_LOGIN_FAILED a ukončí spojení. Pokud se nejedná vůbec o čistě číselnou hodnotu, tak server pošle zprávu SERVER_SYNTAX_ERROR a ukončí spojení.

Syntaktická chyba

Na syntaktickou chybu reaguje server vždy okamžitě po obdržení zprávy, ve které chybu detekoval. Server pošle robotovi zprávu `SERVER_SYNTAX_ERROR` a pak musí co nejdříve ukončit spojení. Syntakticky nekorektní zprávy:

- Příchozí zpráva je delší než počet znaků definovaný pro každou zprávu (včetně ukončovacích znaků `\a\b`). Délky zpráv jsou definovány v tabulce s přehledem zpráv od klienta.
- Příchozí zpráva syntakticky neodpovídá ani jedné ze zpráv `CLIENT_USERNAME`, `CLIENT_KEY_ID`, `CLIENT_CONFIRMATION`, `CLIENT_OK`, `CLIENT_RECHARGING` a `CLIENT_FULL_POWER`.

Každá příchozí zpráva je testována na maximální velikost a pouze zprávy `CLIENT_CONFIRMATION`, `CLIENT_OK`, `CLIENT_RECHARGING` a `CLIENT_FULL_POWER` jsou testovány na jejich obsah (zprávy `CLIENT_USERNAME` a `CLIENT_MESSAGE` mohou obsahovat cokoliv).

Logická chyba

Logická chyba nastane pouze při nabíjení - když robot pošle info o dobíjení (`CLIENT_RECHARGING`) a po té pošle jakoukoliv jinou zprávu než `CLIENT_FULL_POWER` nebo pokud pošle zprávu `CLIENT_FULL_POWER`, bez předchozího odeslání `CLIENT_RECHARGING`. Server na takové situace reaguje odesláním zprávy `SERVER_LOGIC_ERROR` a okamžitým ukončením spojení.

Timeout

Protokol pro komunikaci s roboty obsahuje dva typy timeoutu:

- `TIMEOUT` - timeout pro komunikaci. Pokud robot nebo server neobdrží od své protistrany žádnou komunikaci (nemusí to být však celá zpráva) po dobu tohoto časového intervalu, považují spojení za ztracené a okamžitě ho ukončí.
- `TIMEOUT_RECHARGING` - timeout pro dobíjení robota. Po té, co server přijme zprávu `CLIENT_RECHARGING`, musí robot nejpozději do tohoto časového intervalu odeslat zprávu `CLIENT_FULL_POWER`. Pokud to robot nestihne, server musí okamžitě ukončit spojení.

Speciální situace

Při komunikaci přes komplikovanější síťovou infrastrukturu může docházet ke dvěma situacím:

- Zpráva může dorazit rozdělena na několik částí, které jsou ze socketu čteny postupně. (K tomu dochází kvůli segmentaci a případnému zdržení některých segmentů při cestě sítí.)
- Zprávy odeslané brzy po sobě mohou dorazit téměř současně. Při jednom čtení ze socketu mohou být načteny obě najednou. (Tohle se stane, když server nestihne z bufferu načíst první zprávu dříve než dorazí zpráva druhá.)

Za použití přímého spojení mezi serverem a roboty v kombinaci s výkonným hardwarem nemůže k těmto situacím dojít přirozeně, takže jsou testovačem vytvářeny uměle. V některých testech jsou obě situace kombinovány.

Každý správně implementovaný server by se měl umět s touto situací vyrovnat. Firmware robotů s tímto faktem počítají a dokonce ho rády zneužívají. Pokud se v protokolu vyskytuje situace, kdy mají zprávy od robota předem dané pořadí, jsou v tomto pořadí odeslány najednou. To umožňuje sondám snížit jejich spotřebu a zjednodušuje to implementaci protokolu (z jejich pohledu).

Optimalizace serveru

Server optimalizuje protokol tak, že nečeká na dokončení zprávy, která je očividně špatná. Například na výzvu k autentizaci pošle robot pouze část zprávy s uživatelským jménem. Server obdrží např. 22 znaků uživatelského jména, ale stále neobdržel ukončovací sekvenci `\a\b`. Vzhledem k tomu, že maximální délka zprávy je 20 znaků, je jasné, že přijímaná zpráva nemůže být validní. Server tedy zareaguje tak, že nečeká na zbytek zprávy, ale pošle zprávu `SERVER_SYNTAX_ERROR` a ukončí spojení. V principu by měl postupovat stejně při vyzvedávání tajného vzkazu.

V případě části komunikace, ve které se robot naviguje k cílovým souřadnicím očekává tři možné zprávy: `CLIENT_OK`, `CLIENT_RECHARGING` nebo `CLIENT_FULL_POWER`. Pokud server načte část neúplné zprávy a tato část je delší než maximální délka těchto zpráv, pošle `SERVER_SYNTAX_ERROR` a ukončí spojení. Pro pomoc při optimalizaci je u každé zprávy v tabulce uvedena její maximální velikost.

Ukázka komunikace

```
C: "Oompa Loompa\a\b"
S: "107 KEY REQUEST\a\b"
C: "0\a\b"
S: "64907\a\b"
C: "8389\a\b"
S: "200 OK\a\b"
S: "102 MOVE\a\b"
C: "OK 0 0\a\b"
S: "102 MOVE\a\b"
C: "OK -1 0\a\b"
S: "104 TURN RIGHT\a\b"
C: "OK -1 0\a\b"
S: "104 TURN RIGHT\a\b"
C: "OK -1 0\a\b"
S: "102 MOVE\a\b"
C: "OK 0 0\a\b"
S: "105 GET MESSAGE\a\b"
C: "Tajny vzkaz.\a\b"
S: "106 LOGOUT\a\b"
```

Testování

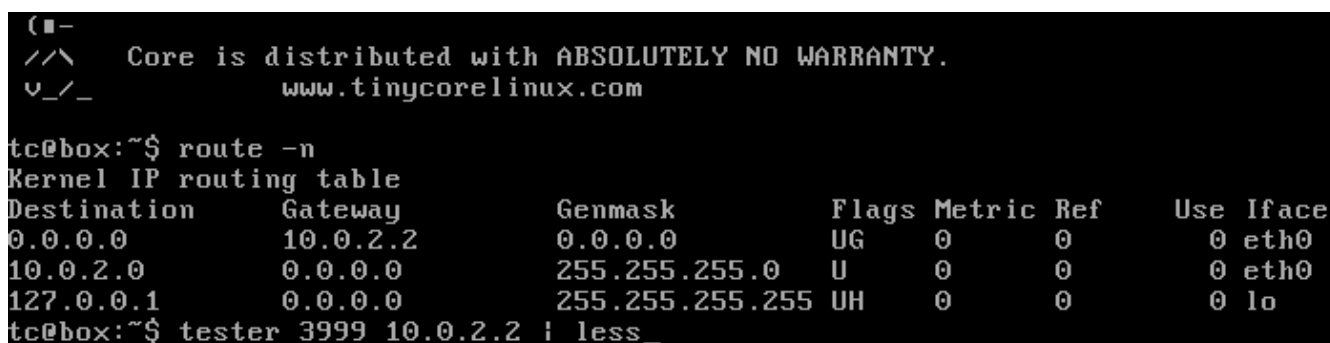
K testování je připraven obraz operačního systému Tiny Core Linux, který obsahuje tester domácí úlohy. Obraz je kompatibilní s aplikací VirtualBox.

Tester

Stáhněte a rozbalte obraz. Výsledný soubor spusťte ve VirtualBoxu. Po spuštění a nabootování je okamžitě k dispozici shell. Tester se spouští příkazem *tester*:

```
tester <číslo portu> <vzdálená adresa> [čísla testů]
```

Prvním parametrem je číslo portu, na kterém bude naslouchat váš server. Následuje parametr se vzdálenou adresou serveru. Pokud je váš server spuštěn na stejném počítači jako VirtualBox, použijte adresu defaultní brány. Postup je naznačen na následujícím obrázku:



```
(■-
/\/\  Core is distributed with ABSOLUTELY NO WARRANTY.
v_/_  www.tinycorelinux.com

tc@box:~$ route -n
Kernel IP routing table
Destination      Gateway          Genmask         Flags Metric Ref    Use Iface
0.0.0.0          10.0.2.2        0.0.0.0         UG    0      0      0 eth0
10.0.2.0         0.0.0.0         255.255.255.0   U     0      0      0 eth0
127.0.0.1        0.0.0.0         255.255.255.255 UH    0      0      0 lo
tc@box:~$ tester 3999 10.0.2.2 | less_
```

Výstup je poměrně dlouhý, proto je výhodné přesměrovat jej příkazem */less*, ve kterém se lze dobře pohybovat, nebo lze použít klávesovou kombinaci "Shift+PageUp" nebo "Shift+PageDown" pro pohyb ve.- výstupu nahoru nebo dolů (historie je však krátká, nelze se posunout moc daleko nazpět).

Pokud není zadáno číslo testu, spustí se postupně všechny testy. Testy lze spouštět i jednotlivě. Následující ukázka spustí testy 2, 3 a 8:

```
tester 3999 10.0.2.2 2 3 8 | less
```

Hodnocení úlohy

Jednotlivé testy, které tester provádí, jsou rozděleny do tematických sekcí. Pro úspěšné získání zápočtu je nutné projít všemi testy z povinné sekcí a lze tak získat 10 bodů k zápočtu. Ostatní části nejsou povinné a lze za ně získat dohromady dalších 10 bodů. Přehled jednotlivých sekcí:

- Ideální situace (povinná)
- Autentizace (povinná)
- Segmentace a spojování (povinná)
- Timeout (povinná)
- Pohyb sondy/robota (povinná)
- Syntaktické chyby
- Optimalizace
- Dobíjení
- Vícevláknovost
- Test náhodnými hodnotami
- Finální test (spustí se paralelně tři instance testu náhodnými hodnotami)

Tester po skončení vypíše protokol o výsledcích.

Možné problémy v operačním systému windows

V některých instalacích OS Windows bývá problém se standardní konfigurací virtuálního stroje. Pokud se nedaří spojit tester ve virtuálce s testovaným serverem v hostitelském operačním systému, tak použijte následující postup:

- U vypnuté virtuálky s testerem změňte nastavení síťového adaptéru z NAT na Host-only network.
- V hostitelském OS by se mělo objevit síťové rozhraní patřící VirtualBoxu. To lze zjistit z příkazové řádky příkazem *ipconfig*. IP adresa tohoto rozhraní bude pravděpodobně 192.168.56.1/24.
- Ve virtuálce s testerem je teď nutné ručně nastavit IP adresu síťovému rozhraní eth0:

```
sudo ifconfig eth0 192.168.56.2 netmask 255.255.255.0
```

- Nyní je možné spustit tester, ale jako cílovou adresu zadejte IP adresu síťového rozhraní v hostitelském OS:

```
tester 3999 192.168.56.1
```

Ke stažení

Aplikace VirtualBox: <https://www.virtualbox.org/wiki/Downloads>

Tester: [Všechny verze testeru ke stažení \(https://drive.google.com/drive/folders/1QzPyZzELNWZhjtbaTGehyNu-zgHclnta?usp=sharing\)](https://drive.google.com/drive/folders/1QzPyZzELNWZhjtbaTGehyNu-zgHclnta?usp=sharing)

Adresář z předchozího linku obsahuje adresáře s vývojovými verzemi testeru. V každém adresáři označeném jako vX (kde X je číslo verze) naleznete následující soubory:

- BI-PSI_tester_2021_vX.ova - virtuálka s testerem
- psi-tester-2021-vX_x64.bz2 - verze pro linux 64-bit
- psi-tester-2021-vX_x86.bz2 - verze pro linux 32-bit

- psi-tester-2021-vX_arm.bz2 - verze pro arm (použitelné pro uživatele MacBook s procesorem M1 - čtěte dále)

Uživatelé MacBooku s procesorem M1 mají trochu komplikovanější spouštění. Společnost Apple bohužel nedává moc možností jak zkompilovat C++ kód kdekoliv jinde, než v operačním systému MacOS. Tester zkompilovaný pro procesory ARM je kompatibilní s architekturou M1, ale protože je zkompilovaný pro linux, nelze jej nativně spustit. Doporučený způsob testování je použít VirtualBox (viz link výše) a v něm nainstalujte libovolnou linuxovou distribuci pro architekturu ARM. Tester pro arm by měl v takto nainstalovaném systému fungovat. (Bohužel nemůžeme připravit image pro ARM, protože nemáme stroj s potřebnou architekturou.)

Požadavky na řešení

- Řešení lze vytvořit v jakémkoliv programovacím jazyce.
- Přijato bude pouze řešení, které projde všemi povinnými testy. Povinné testy jsou ve výpisu testeru označeny jako "mandatory". Tester rovnou spočítá výsledné body (pozor - tester nebere v úvahu časové bonusy a penalizace!).
 - Povinné testy dávají dohromady 10 bodů.

Odevzdání

Úloha je úspěšně odevzdána pouze v případě, že byl zdrojový kód nahrán na odevzdávací server a řešení bylo osobně odprezentováno na laboratorním cvičení! Datum odevzdání je určeno uploadem na odevzdávací server, prezentovat úlohu je tedy možné i po deadline a bez penalizace. Úloha se vypracovává samostatně. Přijímáme pouze originální řešení.



DEADLINE pro upload bez penalizace je 30. 4. 2023!



Po tomto datu bude za každý započatý týden snížen bodový zisk z úlohy o dva. Pro získání zápočtu je pak nutné dodat řešení za více než požadovaných 10 bodů tak, aby se vykompenzovala udělená penalizace.



Úloha musí být odprezentována nejpozději do konce zápočtového týdne!

Předčasné odevzdání bude naopak odměněno body za aktivitu. Každý týden předstihu bude honorován jedním bodem až do maximálně 3 bodů. Harmonogram možných bonusových bodů:

- do 9.4. +3 body

- do 16.4. +2 body
- do 23.4. +1 bod

Pozor, body za aktivitu může získat pouze 100% řešení. Body za aktivitu budou hromadně uděleny (nebo korigovány) až na konci semestru, až bude odevzdávání úloh uzavřeno. Body budou počítány samozřejmě k datu nahrání na archivační server před prezentací.

Ještě uvádíme harmonogram penalizací:

- od 1.5. -2 body
- od 8.5. -4 body
- od 15.5. -6 bodů

Nahrání na archivační server

K odevzdávání slouží speciální server (PSI bouda). **Každý student se na něm registruje a ve vlastním zájmu bude nahrávat svá průběžná řešení, aby byla u každého dohledatelná historie pro případ podezření z plagiarismu.** Při registraci na server je nutné zvolit programovací jazyk. Pokud jej potřebujete změnit, kontaktujte správce serveru (cernyvi2@fit.cvut.cz (<mailto:cernyvi2@fit.cvut.cz>)). Na konci semestru budou všechny odevzdané zdrojové kódy otestovány na duplicitu. V případě shody dvou a více kódů mohou průběžná odevzdání pomoci. Zdrojový kód se nahrává v jednom souboru a nekomprimovaný. Odevzdávací server kód nekontroluje, pouze jej porovnává s kódy ostatních studentů a vyhledává shodu. Je tedy možné spojit více zdrojových kódů do jednoho i když takový kód není potom bez úprav zkompilovatelný.



Pište svůj kód do jediného souboru. Velmi Vám to ulehčí odevzdání.



Pokud máte řešení rozdělené do více souborů, stačí když jejich obsah spojíte do jednoho. Výsledný soubor nemusí být kompilovatelný, ale musí obsahovat veškerý zdrojový kód vašeho řešení. Na linuxu můžete soubory sloučit jednoduchým příkazem: `cat *.java > all.java`

Link na odevzdávací server: [PSI bouda \(https://bouda.fit.cvut.cz\)](https://bouda.fit.cvut.cz)



Pokud jste nenalezli svůj oblíbený jazyk v nabídce, postěžujte si zde: cernyvi2@fit.cvut.cz (<mailto:cernyvi2@fit.cvut.cz>).



Pokud jste nedostali autentizační mail do 24 hodin, postěžujte si zde: cernyvi2@fit.cvut.cz (<mailto:cernyvi2@fit.cvut.cz>).

Osobní prezentace

Prezentace lze výhradně v době laberek a to v sudém i lichém týdnu, pokud vyučující neurčí jinak. Prezentovat lze nejpozději do konce zápočtového týdne. Prezentaci lze provést kdykoliv během cvičení. Na prezentaci se není nutné objednávat. Student během prezentace musí prokázat, že kódu rozumí a že kód funguje. Kód, který je prezentován, musí být stejný jako ten odevzdaný na odevzdávací server. Kontrola probíhá v následujících krocích:

1. Student prokáže svou totožnost.
2. Student ukáže zdrojový kód a spustí test tak, aby bylo zřejmé, že je testován prezentovaný kód.
3. Student odpoví na kontrolní otázky ke zdrojovému kódu.
4. Student nahraje zdrojový kód na odevzdávací server tak, aby bylo zřejmé, že nahrává opravdu prezentovaný kód.

Je na každém studentovi, aby zajistil hladký průběh všech těchto kroků. Pokud nějaký z předchozích nelze splnit, domluvte se svým cvičícím jak postupovat dál.



Prezentujte úlohu co nejdříve po finálním uploadu na server Bouda. Čím déle budete prezentaci oddalovat, tím více hrozí, že zapomenete detaily implementace a nebudete pak dostatečně přesvědčiví, že implementaci rozumíte.