Postup odevzdání semestrální práce:

Semestrální práce se prezentuje výhradně na počítačích a softwarovém vybavení dostupném v učebnách, nelze prezentovat na vlastním notebooku, nebo jiném zařízení, či servery (eryx, ares, ...). Pokud studentům schází nějaká knihovna, nebo verze Javy, je student schopen si ji zajistit bez nutnosti kooperace vyučujícího.

1. Student předvede přeložení jak klienta tak serveru pomocí automatických nástrojů, make, ant, maven, scons (nebo jiným dostupným nástrojem, každopádně bez IDE) na odpovídajících platformách
    a. server: Linux
    b. klient: Linux, Windows
2. Student předvede spuštění programů včetně dostupných parametrů
    a. server: nápověda dostupných parametrů, nastavení adresy pro naslouchání (konkrétní IP adresa, nebo všechny - INADDR_ANY), port a další parametry pro hry dle uvážení studenta
    b. klient: nápověda, adresa a port serveru (v podobě IP, nebo jména), volitelně identifikace rozehrané hry a identifikace hráče a další dle uvážení studenta
3. Student předvede hladký průběh hry (dohraje se hra alespoň dvou hráčů až do konce)
4. Student předvede schopnost aplikace vyrovnat se s problémy na sítí
    a. nedostupný server při startu
    b. nedostupný server během hry (iptables -j DROP), krátkodobě, trvale
    c. nedostupný server během hry (iptables -j REJECT), krátkodobě, trvale
    d. zpoždění při komunikaci a prohazování UDP datagramů - https://github.com/dsiroky/socket_retarder
5. Student předvede schopnost aplikace vyrovnat se s nevalidními daty
    a. server: cat /dev/random | nc server_address server_port
    b. klient: cat /dev/random | nc -l -p server_port
6. Student předvede návrat do hry po přerušení, podle pravidel dané hry jsou následující možnosti:
    a. čeká se návrat hráče do hry
    b. hráč se připojí kdykoli v průběhu hry
    c. hráč se vrátí do hry v dalším kole
7. Student předvede plynulost uživatelského rozhraní nezávislé na zpoždění sítových operací.
8. Student předvede a vysvětlí význam záznamů v logu
9. Student předvede dokumentaci semestrální práce
    a. práce obsahuje základní popis hry
    b. práce obsahuje popis protokolu dostatečný pro implementaci alternativního klienta/serveru, struktura zpráv, datové typy, validace dat, omezení vstupních hodnot, posloupnost nebo návaznost zpráv, ošetření a hlášení chybových stavů

c. práce obsahuje popis implementace klienta a serveru: rozvrstvení aplikace, dekompozice problému, použité knihovny, metoda paralelizace
d. práce obsahuje požadavky na spuštění a běh aplikace
e. práce obsahuje postup překladu jednotlivých částí (ne pomocí IDE)

Průběžné odevzdání

Student v průběhu vypracování práce odevzdává dvakrát průběžnou prezentaci stavu vývoje hry. Odevzdání je bonusové. Vyučující se domluví se studentem na termínech předvedení semestrální práce.

Popis protokolu

Po zvolení hry a použitého transportního protokolu odevzdává student popis vlastního aplikačního protokolu, kterým bude komunikovat server a klient (binární nebo textový protokol). Protokol bude obsahovat strukturu zpráv, přenášené datové typy, návratové kódy, validace a posloupnost kroků.

Odevzdání je možné v elektronicky (.PDF nebo jiný vhodný formát).

Kostra serveru

Student odevzdá a prezentuje kostru/demo serverové části aplikace (ale co prezentuje. Částečnou práci nebo konečné řešení. Oni budou mít co dělat se sebou, s C, Javou, sockety, atd. Jestliže to umí, pak se nebráním. Jinak by mi stačilo předvedení po dohodě z klávesnice s využitím nc) pomocí shell skriptu (bash, ksh, ...), nebo ručně základní implementaci serveru, který odpovídá dle specifikace na volání klienta. Klient je nahrazen jednoduchým shell scriptem nebo studentem za pomoci telnetu, nc a dalších nástrojů.

Požadavky na aplikace

1. Počet hráčů ve hře je omezen pouze pravidly dané hry
2. Počet paralelně hratelných her je nastavitelný při spuštění serveru, případně se použije vhodně zvolený výchozí počet nebo není tento počet omezen, server na plný pool her reaguje pro klienta pochopitelným způsobem (chybová zpráva), omezení není v kódu aplikace napevno, celkový počet her po dobu běhu serveru není omezen, žádné “Už jsme odehráli 4 hry, teď musím restartovat server...”. Student si volí sám vhodný způsob reprezentace her a hráčů, tzn. je možné použít pole o velikosti definované při startu serveru, nebo vhodnější struktury jako spojový seznam atp.
3. Server a klienti viditelně reagují na odpojení hráče
4. Klient nepotřebuje žádné magické informace pro návrat do hry

5. Klient má alespoň základní jednoduché grafické rozhraní zobrazující stav hry a umožňující další tahy ve hře, žádné zadávání karet, tahů typu: 17 = zelená sedma, nebo zelená <enter> sedma <enter>, případně vypiš co mám na ruce<enter>
6. O všech událostech je uživatel informován, připojení k serveru, chyba připojení, chyba serveru, kdo je aktuálně na tahu, jaký byl tah protihráče, chyba klienta, nutnost potvrdit dialog, výzva k provedení nějaké akce
7. Pokud je po uživateli požadována nějaká akce je to dostatečně zřetelné
8. Po skončení hry, je možné dále pokračovat, nelze ukončit klienta a nutit hráče k novému připojení atp.
9. Jednotlivé hry jsou na sobě nezávislé, nelze, že v jedné hře zvolím 5 hráčů a nově příchozí musí toto pravidlo respektovat a čekat až se jich 5 sejde, nelze ani 5 hráčů zakódovat napevno, nebo aby to bylo globálním parametrem serveru, pokud to nedefinují pravidla hry
10. Jak klient tak server běží bez přerušení, všechny vyjímky jsou ošetřené, server nepadá na segfaultech a tak podobně (ať již samovolně, nebo vlivem některé definované chyby na síti nebo v komunikaci)