Študijné materiály pre predmet Základy programovania pre biológov

Contents

Úvod

Kontakt a rozvrh

- Mgr. Broňa Brejová, PhD., kancelária M-163, brejova@fmph.uniba.sk

- Konzultačné hodiny: Utorok 16:00-17:00 alebo po dohode e-mailom.
- Prednáška: Utorok 9:50-11:20 v M-218 na FMFI, miestnosť k dispozícii už od 9:30.

Ciele predmetu

Tento predmet je určený pre študentov biologických a príbuzných odborov. Cieľom je naučiť sa základy práce v operačnom systéme Linux a základy programovania v programovacom jazyku Perl a aplikovať tieto zručnosti na prácu s bioinformatickými nástrojmi.

Študenti dostanú prístup k Linuxovému serveru, na ktorý sa dá pripojiť aj z Windowsových počítačov. Nie je teda potrebné inštalovať Linux na vlastný počítač.

Stručná osnova

Práca v linuxe a na príkazovom riadku, spúšťanie bioinformatických programov, jednoduché spracovanie súborov v textovom formáte, UCSC genome browser a iné zdroje dát, základy programovacieho jazyka Perl.

Pravidlá

- Hodnotenie: A: 90+, B: 80+, C: 70+, D: 60+, E: 50+
- Domáce úlohy (protokoly): 60%
- Záverečný projekt a ústna skúška: 40%

Na tomto predmete sa budeme stretávať na dve vyučovacie hodiny týždenne v počítačovej učebni. Približne polovicu času bude tvoriť prednáška/demonštrácia a v druhej polovici času začnete individuálne (s pomocou vyučujúcej) pracovať na **domácej úlohe** súvisiacej s prednáškou. Čo nestihnete, budete mať za úlohu dokončiť doma, resp. v počítačovej učebni. Z domácej úlohy treba spísať stručný **protokol** (poznámky). Na dokončenie protokolu máte vždy dva týždne od príslušnej hodiny. Účasť na hodine nie je povinná, ale treba odovzdať protokoly aj z vymeškaných hodín. V prípade, že sa nemôžete zúčastniť zo závažných dôvodov, napríklad kvôli chorobe, kontaktujte vyučujúcu.

40% známky bude tvoriť malý **projekt**, ktorý si sami vyberiete a vypracujete v podobnom formáte ako protokoly. Projekt sa bude odovzdávať počas skúškového obdobia do určeného termínu. Po oznámkovaní projektu sa bude konať ešte krátka ústna skúška týkajúca sa projektu, ktorá môže ovplyvniť vašu výslednú známku. Viac o projektoch na zvláštnej stránke.

Tento predmet **nie je určený pre študentov informatických odborov** (informatika, aplikovaná informatika, kognitívna veda, učiteľstvo informatiky a pod.). Ak si ho takíto študenti zapíšu, budú musieť okrem pravidelných protokolov odovzdať aj **rozsiahlejší projekt** naprogramovaný v jazyku Perl na spracovanie biologických dát. Váha tohto projektu bude 60%, domáce úlohy sa preškálujú na 40%.

Protokoly

Protokoly sa píšu elektronicky v systéme Moodle a netreba ich odovzdávať v papierovej podobe. Ich hlavnou zložkou je **postup**, akým ste príklad vyriešili (napríklad príkazy, ktoré ste spustili, alebo program, ktorý ste napísali) a aký ste dostali **výsledok**. Za tieto základné informácie môžete získať 60-80% bodov, podľa typu domácej úlohy. Zvyšné body môžete získať tým, že si spíšete **poznámky** vysvetľujúce, prečo ste úlohu riešili práve takto a ak vaše prvé pokusy boli neúspešné, môžete zdokumentovať aj tie. Protokol má dva ciele: preukázať, že ste príklad vyriešili a tiež spoznámkovať si postup a prípadné úskalia pre vašu vlastnú potrebu, ak by ste podobný problém potrebovali riešiť niekedy v budúcnosti, keď si už detaily nebudete pamätať.

Opisovanie

V protokoloch môžete použiť úryvky textu z materiálov k predmetu prípadne aj z iných zdrojov, je treba však vždy uviesť, z akého zdroja pochádzajú. Všetok zvyšný text by ste mali napísať vlastnými slovami. U domácich úloh povoľujeme a podporujeme diskusiu medzi študentami, ale protokol by mal každý študent vypracovať samostatne na základe samostatne vykonanej práce.

Opisovanie domácich úloh a projektov je vážnym porušením akademickej integrity. Opisovanie domácich úloh a projektov zahŕňa opísanie práce inej osoby (napr. od spolužiaka, z internetu, alebo z odbornej literatúry) a jej odovzdanie pod vlastným menom, umožnenie inej osobe opísať vlastné riešenia, ale aj spoluprácu pri riešení nad rámec povolený prednášajúcimi predmetu. Pri zistení opisovania budeme štandardne prideľovať známku -100% za príslušnú časť práce, s minimálnym znížením výslednej známky o jeden stupeň. Obzvlášť hrubé porušenia akademickej integrity budú postúpené na riešenie dekanovi fakulty.

Projekt

- Termín odovzdania návrhu na projekt: utorok 14.1. do 23:55
- Termín odovzdania projektov: pondelok 17.1.2011 do 23:55.
- Odovzdávanie projektu (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle /mod/assignment/view.php?id=15)
- Odovzdávanie projektu - alternatívne ako pdf súbor (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=25)
- Odovzdávanie návrhu na projekt (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle /mod/assignment/view.php?id=24)

Námety na projekt

- Naučiť sa viac o HTML a vytvoriť webstránku, pokiaľ možno s obsahom týkajúcim sa biológie alebo bioinformatiky (detaily na konci prednášky 4).
- Preskúmať niektorý z iných genome browserov (detaily na konci prednášky 3).
- Spustiť nejaké programy na niektorom celom genóme a z výsledkov spraviť UCSC genome browser custom track aj s vysvetlivkami
- Naučiť sa pracovať s relačnými databázami a jazykom SQL (dá sa pracovať napr. s tabuľkami v UCSC genome browseri)
- Naučiť sa pracovať s programom R na štatistické výpočty a zobrazovanie dát. Viac informácií tu.
- Naučiť sa viac o jazyku Perl, napríklad používať knižnicu Bioperl, ktorá vie pracovať s rôznymi formátmi používanými v biológii (detaily na konci prednášky 10).
- Aplikovať veci preberané na predmete na dáta, ktoré vás obzvlášť zaujímajú

Projekty

Projekt by mal byť rozsahom ekvivalentný zhruba 3 domácim úlohám. Je pre vás možnosťou ako sa viac zaoberať tou časťou predmetu, ktorá vás najviac zaujala. Je niekoľko možných typov projektu:

1. Vyberiete si niekoľko cvičení z domácich úloh a skúsite nejaké ich obmeny (napr. spustiť ten istý program na iných dátach alebo s inými nastaveniami).
2. Zoženiete si nejaké reálne dáta (napr. súvisiace s vašou diplomovu prácou alebo voľne prístupné na internete) a skúsite na nich spraviť nejakú bioinformatickú analýzu s použitím techník, ktoré sme sa učili.
3. Naštudujete si nejakú ďalšiu techniku, ktorú sme nepreberali a jej použitie si vyskúšate na niekoľkých jednoduchých cvičeniach.

O téme projektu a postupe sa môžete priebežne radiť s vyučujúcou. Doporučujem začať rozmýšľať o téme projektu pomerne skoro, nenechávať to až na koniec semestra.

Správa z projektu

Projekt odovzdávajte vo forme správy, ktorá by mala obsahovať:

- krátky úvod vysvetľujúci ciele projektu
- poznámky k tomu, čo ste sa pri práci na projekte naučili (obzvlášť pre projekty, kde si naštudujete nový materiál, napr. jazyk HTML)
- protokol, obsahujúci príkazy, ktoré ste používali, prípadne iné použité postupy, výsledky programov a podobne
- záver, v ktorom zhrniete, čo sa vám podarilo na projekte dosiahnuť, v čom ste naopak narazili na problémy a prípadne nejaký význam projektu pre vašu ďalšiu prácu

Správu z projektu môžete písať priamo v systéme Moodle, podobne ako protokoly k domácim úlohám, alebo ju môžete vypracovať napr. v MS Worde, vyexportovať do pdf a odovzdať pdf súbor. Dáta, program a iné veci, ktoré ste na projekte vytvorili, uložte niekam na server vyuka a meno adresára uveďte v správe.

Návrh projektu

Ako špeciálnu domácu úlohu treba do 14.12. odovzdať prostredníctvom systému Moodle krátky návrh projektu (jeden odstavec textu). Popíšte, čo by ste chceli dosiahnuť, aké informatické prostriedky plánujete použiť, čo sa chcete naučiť a pod. Cieľom je, aby som vám k návrhu mohla napísať komentáre, ktoré vám môžu pomôcť pri realizácii projektu, prípadne doporučím zmenu témy. Preto sa oplatí tento návrh odovzdať čím skôr. Ak narazíte na ťažkosti, nemusíte sa svojho návrhu za každú cenu držať, radikálne zmeny v smerovaní projektu však doporučujem vopred konzultovať s vyučujúcou.

Harmonogram prednášok

Týždeň 20.-24.9.

Úvod, organizácia predmetu, čo je to linux, príkazový riadok

Prednáška 1 · Domáca úloha 1 (5%) · Odovzdávanie D.Ú.1 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=12) · Softvér na stiahnutie

Týždeň 27.9.-1.10.

Prihlásenie sa na server z Windows a Linuxu, práca na príkazovom riadku, pohyb v adresároch

Prednáška 2 · Domáca úloha 2 (6%) · Odovzdávanie D.Ú.2 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=13)

Týždeň 4.-8.10.

Kopírovanie súborov v Linuxe, UCSC genome browser, získavanie dát

Prednáška 3 · Domáca úloha 3 (6%) · Odovzdávanie D.Ú.3 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=14)

Týždeň 11.-15.10.

Práca so súbormi v Linuxe, jednoduchá webstránka

Prednáška 4 · Domáca úloha 4 (6%) · Odovzdávanie D.Ú.4 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=16)

Týždeň 18.-22.10.

Programovanie v jazyku Perl: jednoduché spracovanie súboru

Prednáška 5 · Domáca úloha 5 (5%) · Odovzdávanie D.Ú.5 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=17)

Týždeň 25.-29.10.

Programovanie v jazyku Perl: polia

Prednáška 6 · Domáca úloha 6 (3%) · Odovzdávanie D.Ú.6 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=18)

Týždeň 1.-5.11.

Rektorské voľno

Týždeň 8.-12.11.

Programovanie v jazyku Perl: základné pojmy (premenná, cyklus, podmienka)

Prednáška 7 • Domáca úloha 7 (5%) • Odovzdávanie D.Ú.7 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=19)

Týždeň 15.-19.11.

Dekanské voľno, promócie

Týždeň 22.-26.11.

Príkazy na spracovanie textových súborov v Linuxe, spúšťanie bioinformatických programov

Prednáška 8 • Domáca úloha 8 (6%) • Odovzdávanie D.Ú.8 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=20)

Týždeň 29.11.-3.12.

Spúšťanie bioinformatických programov

Prednáška 9 • Domáca úloha 9 (5%) • Odovzdávanie D.Ú.9 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Týždeň 6.-10.12.

Programovanie v jazyku Perl: práca so súbormi, spúšťanie externých programov

Prednáška 10 • Domáca úloha 10 (5%) • Odovzdávanie D.Ú.10 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=22)

Týždeň 13.-17.12.

Programovanie v jazyku Perl: asociatívne polia, súbory

Prednáška 11 • Domáca úloha 11 (3%) • Odovzdávanie D.Ú.11 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=23) • Odovzdávanie D.Ú.12 plán projektu (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Poďakovanie: Časť materiálov na týchto stránkach pomohli pripraviť Tomáš Vinař a Tina Višňovská.

Softvér na stiahnutie

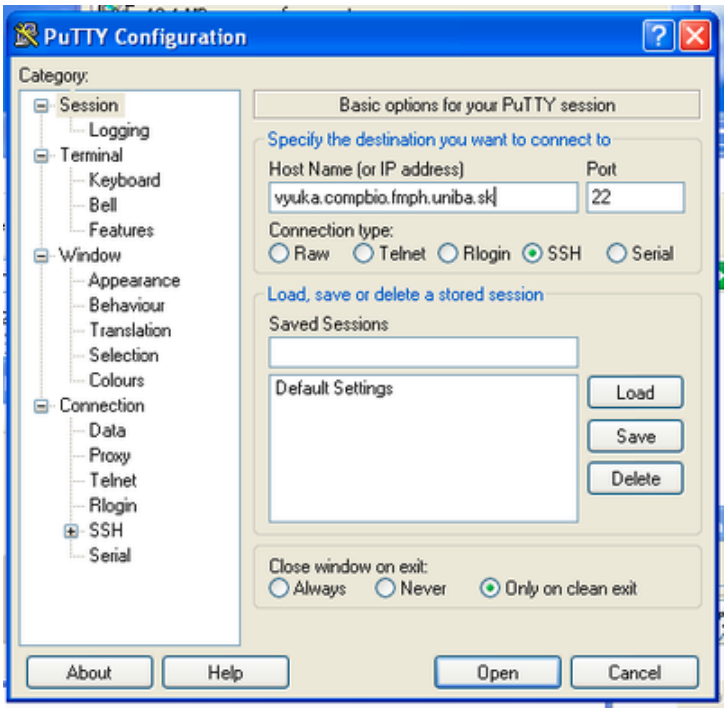
Na tomto predmete budete pracovať na serveri vyuka.compbio.fmph.uniba.sk. Ak chcete pracovať na domácich úlohách zo svojho počítača, potrebujete si nainštalovať softvér, pomocou ktorého sa na server pripojíte.

- Program PuTTY si stiahnete tu (http://www.chiark.greenend.org.uk/%7Esgtatham/putty/)
- Program WinSCP (http://winscp.net/eng/docs/introduction) si stiahnite tu (http://winscp.net/download/winscp425setup.exe)
- Program Xming (http://www.straightrunning.com/XmingNotes/) si stiahnite tu (http://sourceforge.net/projects/xming/files/Xming/6.9.0.31/Xming-6-9-0-31-setup.exe/download)

Ak potrebujete s ich používaním alebo inštaláciou na váš notebook pomôcť, prineste si notebook na hodinu (druhý týždeň semestra, prípadne neskôr).

Používanie programu PuTTY

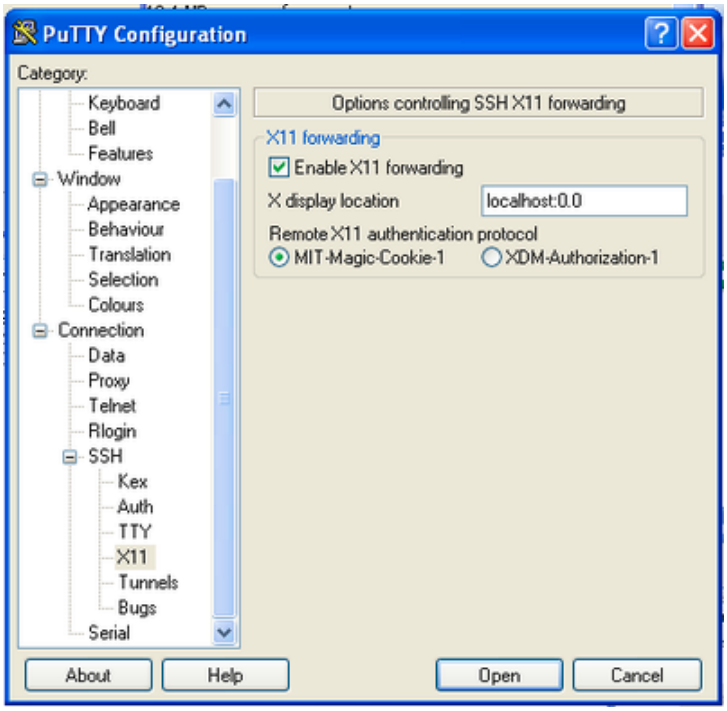
Potom ako získate konto na serveri vyuka.compbio.fmph.uniba.sk (bude vám oznámené na druhej prednáške), môžete sa pripojiť na server programom PuTTY. Zadáme:



Pri prvom prihlásení môžete dostať hlášku "PuTTY Security Alert: The server's host key is not cached ... " Stlačte Yes.

Používanie programu Xming

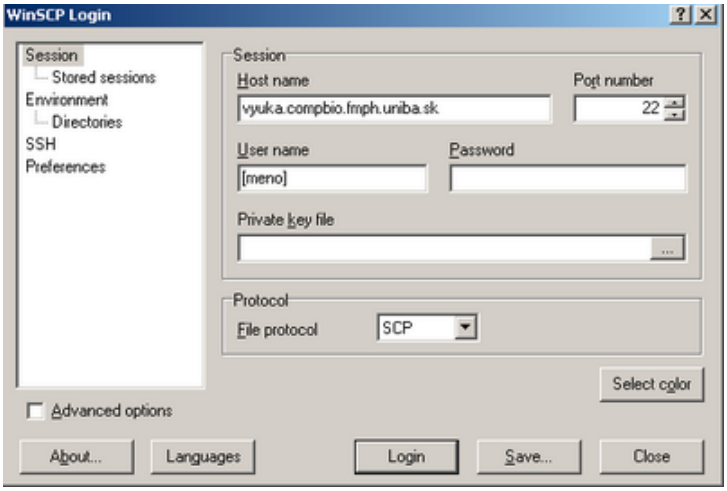
Ak chcete používať grafické programy zo servera (editory, zobrazovače fylogenetických stromov a pod.), potrebujete si nainštalovať aj program XMing. Najskôr spustite XMing, ktorý zdanlivo nič nespraví, iba v pravo dole na lište pridá ikonku veľkého X. Keď už XMing beží, na server sa opäť pripájate cez PuTTY, ale treba pridať jedno nové nastavenie v časti Connection/SSH/X11:



Nastavenia si tiež môžete uložiť, aby ste to nemuseli vždy robiť znovu.

Používanie programu WinSCP

Pomocou programu WinScp môžete prenášať súbory zo servera na Váš počítač a naopak. Keď budete pracovať v učebni na systéme Linux, budete namiesto toho používať program scp. Na pripojenie pomocou WinSCP treba zadať host name a username:



Prednáška 1

Úvod do predmetu, organizačné pokyny

- Webstránka **http://vyuka.compbio.fmph.uniba.sk/ppb/**
- Pozri Úvod a Pravidlá

Linux

- **Operačný systém**: základný program bežiaci na počítači, poskytuje prostredie pre ďalšie aplikácie a užívateľa, napr. Microsoft Windows
- **Linux**: operačný systém, založený na skorších UNIXových operačných systémoch
- Je to slobodný softvér, takže každý si ho môže zadarmo nainštalovať, ale aj meniť a ďalej šíriť
- Začiatky 1991 Linus Torvalds, ďalší vývoj veľkým množstvom programátorov na celom svete (dobrovoľníci aj firmy)

Príkazový riadok

Užívateľské prostredia:

Príkazový riadok	**Grafické prostredie**

▪ S počítačom komunikujeme písaním príkazov, ktoré počítač vykonáva
▪ Veľké množstvo príkazov:
  ▪ ťažšie sa naučiť pre začiatočníkov
	▪ Komunikácia pomocou myši, menu, okien
	▪ Výber z menu:
	  ▪ ľahšie pre začiatočníkov

■ flexibilné a rýchle pre skúsených užívateľov	■ zdĺhavé klikanie pre skúsených užívateľov

Oba typy prostredia sú vo Windows aj Linuxe, ale v Linuxe má príkazový riadok centrálnejšie miesto.

Ďalšie výhody príkazového riadku

- umožňuje spúšťať série príkazov vo forme automatických **skriptov**, z jednoduchých programov zostaviť zložitejší celok
- dlhšie bežiace príkazy môžeme **spustiť na pozadí**, kým robíme niečo iné alebo sa odhlásime z počítača
- rýchlejší prístup k serverom cez sieť (menej prenášanej informácie)
- zvyčajne je jednoduchšie naprogramovať program pre príkazový riadok

Preto **veľa nástrojov v bioinformatike** má túto formu.

Príklad

Príkaz `ls` vypíše zoznam súborov v aktuálnom adresári

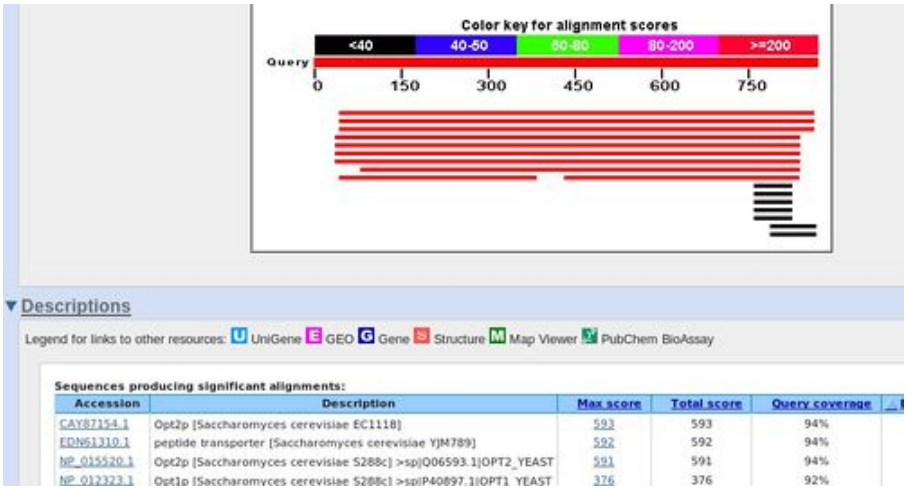
```
ls
```

Zloženie viacerých príkazov - spočítajme súbory, ktorých meno obsahuje písmeno a (podrobnosti neskôr)

```
ls | grep a | wc -l
```

Ukážka použitia Linuxu a Perlu

- Predstavme si, že máme 50 proteínov o ktorých chceme zistiť niečo viac, napr, čo robí ich najbližší homológ z kvasinky Saccharomyces cerevisiae
- Môžeme robiť ručne jeden po druhom na http://blast.ncbi.nlm.nih.gov/



- Alebo si stiahneme všetky proteíny z Saccharomyces cerevisiae, napr. tu [1] (http://downloads.yeastgenome.org/sequence/genomic_sequence/orf_protein/)
- Potom spustíme blast

```
formatdb -i yeast-prot.fa
blastall -p blastp -i prots.fa -d yeast-prot.fa -m 9 -e 0.01 > blastp.out
```

- Výsledky z blastu spracujeme krátkym programom v Perle do tabuľky, ktorú si môžeme otvoriť napr. v Exceli

```
./spracuj.pl yeast-prot.fa blastp.out > vysledok.csv
```

- Súbory sú v adresári `/projects/data-ppb/ukazka/`
- Ak náš zoznam 50 proteínov niekedy zmeníme, za pár sekúnd vieme vytvoriť novú tabuľku.

Cvičenia 1

Prednáška 1 • Odovzdávanie D.Ú.1 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=12)

Dnešné cvičenie má dva ciele:

- Prihlásiť sa na Linuxový počítač v učebni a trochu sa tam rozhliadnuť
- Prihlásiť sa do systému Moodle a odovzdať domácu úlohu.

Nemusíte písať protokol z cvičenia, iba odpovedať na otázky uvedené nižšie. Ak chcete, môžete si do domácej úlohy pridať nejaké poznámky o tomto cvičení pre vlastnú potrebu.

Postup práce s počítačom v učebni

- Zapnite počítač a monitor, v menu zvoľte možnosť Bioinformatika.
- Keď nabehne grafické prihlasovacie okienko, zvoľte užívateľa Bioinformatika a heslo, ktoré ste dostali.
- Na dolnej lište si môžete vybrať často používané programy, napr. internetový prehliadač Firefox.
- Ikonka s čiernou obrazovkou spustí príkazový riadok, skúste si na ňom zadať príkaz `ls`
- Na príkazovom riadku sa na kopírovanie textu nepoužíva Ctrl-C, Ctrl-V, ako ste zvyknutí, ale priamo po vysvietení myšou je už text skopírovaný a stredným tlačítkom myši sa vloží. Neskôr v semestri budete potrebovať kopírovať príkazy a výsledky programov do protokolov.
- V prehliadači si pozrite stránku predmetu http://vyuka.compbio.fmph.uniba.sk/ppb/ a z nej prejdite cez linku na domácu úlohu do systému Moodle
- V systéme Moodle zadajte meno a heslo, ktoré sme vám dali. Heslo si môžete zmeniť.
- V domácej úlohe 1 odpovedzte na otázky uvedené nižšie. Doporučujem skopírovať si potrebnú časť zadania do moodlu, potom vpisovať odpovede na jednotlivé otázky.
- Pred odchodom z miestnosti kliknite na ikonku v pravom dolnom rohu, v menu zvoľte Shut Down, potom potvrďte End current session.
- Počítač by sa mal vypnúť sám, vypnite monitor.

V systéme Moodle napíšte odpovede na nasledujúce otázky, ktoré nám uľahčia Vás kontaktovať a plánovať tento predmet:

- Meno a priezvisko, e-mailovú adresu
- Študijný program a ročník, prípadne nepovinne aj tému a školiteľa diplomovej/bakalárskej práce
- Pracovali ste už niekedy s Linuxom? Ak áno, v akom rozsahu?
- Učili ste sa už niekedy programovať? V akom jazyku a v akom rozsahu?
- Pracovali ste už s nejakými bioinformatickými nástrojmi a databázami? Akými napríklad?
- Je niečo, čo by ste obzvlášť chceli na tomto predmete naučiť?

V riešení tejto domácej úlohy si tiež vyskúšajte možnosti editora v prostredí Moodle. Skúste napríklad nejakú časť textu napísať hrubým písmom a skúste do textu pridať odkaz na nejakú webstránku (pomocou tlačítka s obrázkom reťaze).

Prednáška 2

Prihlásenie, odhlásenie, zmena hesla

- Na tomto predmete budeme pracovať na Linuxovom serveri vyuka.compbio.fmph.uniba.sk
- Ak sa chcete naňho pripojiť z windowsového počítača, pozrite tieto pokyny
- Pripojenie na server z linuxového príkazového riadku (napr, z učebne M218):

```
ssh -XC meno@vyuka.compbio.fmph.uniba.sk
```

- Namiesto `meno` dajte vaše AIS2 prihlasovacie meno (priezvisko + číslo)
- Zmena hesla:

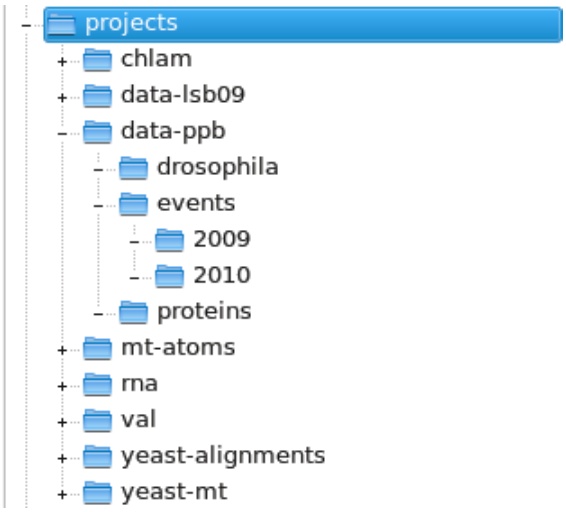
```
passwd
```

- Počítač si vypýta staré heslo, potom 2x nové
- Odhlásenie: príkaz `exit` alebo stlačte Ctrl-d (prípadne zavrite celé okno)

Súbory a adresáre

- Obrázky, texty, dáta a pod. sú uložené v súboroch
- Súbory sú združené v adresároch (priečinkoch) na sprehľadnenie
- Adresár môže obsahovať aj ďalšie adresáre (stromová štruktúra)

Napríklad, takto vyzerá štruktúra niekoľkých adresárov na serveri:



Grafické programy na prácu s adresármi a súbormi

- podobné na prácu vo Windows, presúvanie, mazanie, kopírovanie, atď
- napríklad program dolphin (alebo nautilus, konqueror a ďalšie)
- textový program mc
- naučíme sa však, aj ako robiť tieto veci priamo na príkazovom riadku, dá sa spájať s inými príkazmi

Pohyb po adresároch (ls, cd)

- Jeden adresár je vždy aktuálny, zobrazuje sa nam na prikazovom riadku
- Zoznam súborov a adresárov v aktuálnom adresári dostaneme príkazom `ls`
- Zoznam súborov v nejakom inom adresári dostaneme príkazom `ls iny_adresar`
- Príkaz `cd novy_adresar` zmení aktuálny adresár na zadaný nový adresár

Príklad: keď sa prihlásime, sme v adresári /home/meno. Pomocou príkazu cd sa presunieme do adresára /projects/data-ppb/ (počítač nič nevypíše, iba zmení aktuálny adresár) a skúsime ls. Príkaz ls vypíše obsah adresára data-ppb/: podadresáre 2010-L, events a ukazka a súbor README. Nakoniec vypíšeme obsah podadresára events, ktorý obsahuje ďalšie podadresáre 2009 a 2010.

```
meno@vyuka:~$ cd /projects/data-ppb/
meno@vyuka:/projects/data-ppb$ ls
2010-L  events  README  ukazka
meno@vyuka:/projects/data-ppb$ ls events
2009  2010
```

Absolútne a relatívne cesty

- **Absolútna cesta** určuje, ako sa k danému súboru alebo adresáru dostať z **koreňa**
- Napr. /projects/data-ppb/, /projects/data-ppb/README, /projects/data-ppb/events /2009/Wed_25_Nov_2009, /home/meno/ atď
- Jednotlivé adresáre v strome sa oddeľujú lomítkom
- Absolútna cesta začína lomítkom

- **Relatívna cesta** určuje, ako sa k danému súboru dostať z aktuálneho adresára
- Napr. ak aktuálny adresár je /projects/data-ppb/, tak relatívna cesta k /projects /data-ppb/README je len README a relatívna cesta k /projects/data-ppb/events

/2009/Wed_25_Nov_2009 je events/2009/Wed_25_Nov_2009
- Relatívna cesta nezačína lomítkom
- Relatívna cesta môže ísť aj hore do nadadresára, pomocou ..
- Napr. ak aktuálny adresár je /projects/data-ppb/events/2009, tak relatívna cesta ../../proteins nám dá to isté ako /projects/data-ppb/proteins

V príkazoch ls, cd a ďalších môžeme používať absolútne aj relatívne cesty.

Dôležité adresáre

- **Koreň** je adresár s cestou /, počiatočný bod stromovitej štruktúry adresárov
- **Domovský adresár** (home directory) /home/meno je aktuálny po prihlásení
  - Tam si ukladáme väčšinu svojich súborov, ak nie je dobrý dôvod ich uložiť inde
  - Skratka pre domovsky adresár je ~, napr. `cd ~` v8s tam presunie

Hviezdičková konvencia

- Príkaz ls namiesto všetkých súborov v adresári môže vypísať len tie, ktoré si vyberieme
- Všetky súbory v aktuálnom adresári začínajúce písmenom x vypíšeme pomocou `ls x*`
- Všetky súbory obsahujúce písmeno x hocikde v mene vypíšeme pomocou `ls *x*`

Prezeranie obsahu súboru (less)

- `less [subor]`
- Vypíše súbor na obrazovku, môžeme v ňom listovať pomocou medzery alebo Page up a Page down, vyskočíme pomocou q (ako quit), ďalšie klávesy sa dozviete po stlačení h (help)

Zjednodušenie práce

Užitočné pomôcky na príkazovom riadku:

- kláves Tab
  - ak je len jeden súbor alebo adresár, ktorý pasuje na rozpísaný začiatok slova, doplní ho automaticky
  - ak je súborov alebo adresárov viac, doplní, čo majú spoločné, po opakovanom stlačení ponúkne možnosti
- šípky hore/dole
  - prechádzanie históriou spustených príkazov
- copy&paste myšou
  - ľavým tlačidlom a ťahaním po texte označíme
  - kliknutím stredného tlačidla (kolieska) vložíme kam potrebujeme
  - ak nemáme stredné tlačidlo, klikneme naraz pravým aj ľavým

Ďalšie materiály

- Tutoriál Linuxu (http://www.fsid.cvut.cz/cz/U201/linux.html)
- a ešte jeden (http://tldp.org/LDP/gs/node5.html)
- Viac o hviezdičkovej notácii (http://www.tuxfiles.org/linuxhelp/wildcards.html)

Cvičenia 2

- Odovzdávanie D.Ú.2 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=13)

V tejto domácej úlohe si vyskúšame prihlasovanie na server a rôzne príkazy na prácu so súbormi z druhej prednášky. Doporučujem si otvoriť tri taby alebo okná prehliadača: jeden s prednáškou, jeden so zadaním, jeden s odovzdávacím okienkom v moodli. Zadanie si môžete skopírovať do moodlu a dopĺňať odpovede. Nezabudnie občas odpoveď v moodli uložiť, aby ste náhodou neprišli o veľký kus práce.

Ak nie je uvedené inak, v každej časti A,B,... napíšte príkaz, aký ste použili, a ak počítač niečo vypísal, tak čo vypísal. Môžete si tiež napísať ďalšie poznámky, napr. ak vám na prvýkrát nebolo niečo jasné.

- A. Prihláste sa na server pomocou príkazu ssh
  - Do protokolu skopírujte presnú formu príkazu (s vašim menom) a čo server vypísal po prihlásení
- Ak máte záujem, zmeňte si heslo príkazom passwd
  - Kvôli bezpečnosti vyberajte heslá, ktoré sa nedajú ľahko uhádnuť (nepoužívajte iba jedno slovo)
  - Tento krok nepíšte do protokolu (rozhodne tam nikdy nepíšte žiadne heslá)
- B. Zmeňte aktuálny adresár na /projects/data-ppb/events
- C. Spustite program `dolphin .` (bodka znamená aktuálny adresár) a zistite, aké má adresár events podadresáre a koľko súborov je v podadresári 2010. Odpoveď napíšte do protokolu.
- D. Čo sa stane, ak spustíte program dolphin bez bodky? Aký adresár vám zobrazí?
- E. Späť na príkazovom riadku: presuňte sa späť do adresára /projects/data-ppb/events/ (ako to spravíte?). Potom vyskúšajte si pozrieť obsah nasledujúcich adresárov pomocou príkazu ls, pričom vždy spustite ls dvakrát, s absolútnou aj relatívnou cestu (vzhľadom na aktuálny adresár /projects/data-ppb/events):
  - 2010, data-ppb
- F. Pomocou príkazu `less` si pozrite obsah aspoň jedného súboru v adresári `/projects/data-ppb/events/2010`
- G. Pomocou hviezdičkovej konvencie vypíšte všetky súbory v adresári /projects/data-ppb/events/2009 ktoré obshaujú slovo May v názve (sú to teda akcie, ktoré sa stali v máji)

Prednáška 3

Práca s Linuxom, kopírovanie súborov (cp, scp)

Opakovanie

- na server sa prihlásime príkazom `ssh -XC meno@vyuka.compbio.fmph.uniba.sk`
- zoznam súborov v adresári si pozrieme príkazom `ls meno_adresara`
- obsah textového súboru si môžete pozrieť príkazom `less meno_suboru`
- do iného adresára sa presunieme príkazom `cd meno_adresara`
- mená adresárov a súborov zadávame pomocou absolútnej alebo relatívnej cesty. Absolútna začína lomítkom a ide od koreňa celého systému súborov na počítači.

Relatívna nezačína lomítkom a ide z aktuálneho adresára

Kopírovanie súborov (cp)

- `cp odkiaľ kam`
- Skopíruje súbor `odkiaľ` na miesto `kam`
- Môžeme použiť absolútne alebo relatívne cesty.
- Cieľové miesto `kam` môže byť adresár alebo celé meno súboru

Príklad: ak aktuálny adresár je /projects/data-ppb/, nasledujúce tri príkazy všetky kopírujú súbor README do poadresára events:

```
# relatívne cesty
cp README events/
# absolútne cesty
cp /projects/data-ppb/README /projects/data-ppb/events/
# celé meno súboru
cp README events/README

# tento príkaz kopíruje do /projects/data-ppb/events/README2
cp README events/README2
# ak sa presunieme do adresára events, možeme kopírovať súbor do aktuálneho adresára .
cd events
cp ../README .
```

Kopírovanie súborov zo servera/na server (scp)

- Kríženec medzi ssh a cp (Secure CoPy)
- Na server (spustite na vašom linuxovom počítači napr. v učebni): `scp subor meno@vyuka.compbio.fmph.uniba.sk:nove_meno_suboru`
- Zo servera: `scp meno@vyuka.compbio.fmph.uniba.sk:subor nove_meno_suboru`
- Ak chcete kopírovať súbory medzi serverom a Windowsovým počítačom, nainštalujte si program WinSCP

```
#skopíruje súbor README2 do adresára /projects/data-ppb/ na serveri
scp README2 hrasko37@vyuka.compbio.fmph.uniba.sk:/projects/data-ppb/

#skopíruje súbor README2 do domovského adresára užívateľa hrasko37 na serveri
scp README2 hrasko37@vyuka.compbio.fmph.uniba.sk:

#skopíruje súbor README2 do domovského adresára užívateľa hrasko37 pod menom README3
scp README2 hrasko37@vyuka.compbio.fmph.uniba.sk:README3

#skopíruje súbor README2 z domovského adresára užívateľa hrasko37 na serveri do aktuálneho adresára
scp hrasko37@vyuka.compbio.fmph.uniba.sk:README2 .
```

Upozornenie: dvakrát meraj, raz rež

- Príkazový riadok spraví, čo napíšete, nepýta sa, či to myslíte naozaj
- Príkazy cp a scp môžu prepísať už existujúce súbory
- Neexistuje undo
- Preto si vždy dobre premyslite, čo chcete spraviť a skontrolujte príkaz pred tým, ako dáte Enter

UCSC genome browser

- Niekoľko eukaryotických genómov: http://genome.ucsc.edu/

- Prokaryotické genómy: http://microbes.ucsc.edu/
- Spája príjemné webové klikacie prostredie a možnosť exportovania súborov na ďalšie spracovanie

Prezeranie genómu

- Pozrime si http://genome.ucsc.edu/, zvoľme Genomes na modrom pruhu
- Zvoľme Clade Insect, genome D. melanogaster, position chrX:15700001-15800000
- V dolnej časti stránky môžeme zapínať a vypínať rôzne "tracky"
- V hornej časti sa môžete pohybovať po genóme, približovať a vzďaľovať

Sťahovanie DNA sekvencie

- Pomocou linky DNA v hornom modrom menu si vieme stiahnuť sekvenciu aktuálne zobrazeného úseku vo fasta formáte, potom môžeme uložiť do súboru

Práca s tabuľkami, sťahovanie anotácií

- Položka Tables na hornej lište umožnuje robiť rafinované veci s tabuľkami, ktoré obsahujú súradnice génov a pod.
- Základná vec: vyexportovať napr. všetky gény v zobrazenom výseku v niektorom formáte:
  - sequence: fasta súbor proteínov, génov alebo mRNA s rôznymi nastaveniami
  - GTF: súradnice
  - Hyperlinks to genome browser: klikacia stránka
- Namiesto exportu si môžeme pozrieť rôzne štatistiky

Pokročilejšia práca s tabuľkami

- Zložitejšie: prienik dvoch tabuliek, napr. gény, ktoré sú viac než 50% pokryté simple repeats
  - V intersection zvolíme group: Variation and repeats, track: RepeatMasker, nastavíme All FlyBase Genes records that have at least 50% overlap with RepeatMasker
  - V summary/statistics zistíme, že v genóme ich je 30, môžeme si ich preklikať cez Hyperlinks to genome browser

- Filter na tabuľku, napr. gény, ktoré majú v názve ribosomal
  - Vo filter stránke v Linked tables začiarkneme FlyBaseToDescription
  - Pribudne políčko dm3.flyBaseToDescription based filters, kde do val does match dáme `*ribosomal*`
  - Je 9 takých génov, napr. Putative 60S ribosomal protein L33.

Porovnávanie genómov

- Track Conservation ukazuje, ako dobre sú jednotlivé úseky evolučne konzervované s inými druhmi drosophíl a hmyzu
- Keď sa priblížime na 10Kb, môžeme kliknutím na tento pruh vidieť zarovnanie
- Zapnime D. yakuba Net. Ten vám ukazuje veľké preusporiadanie medzi genómami D.mel. a D.yak.
- Keď klikneme na určitý obdĺžnik v tomto pruhu, vieme sa presunúť na príslušnú časť D.yak. genómu

Ďalšie materiály

- Návody na používanie UCSC browsera nájdete na ich stránke: http://genome.ucsc.edu/training.html

Námet na projekt

Okrem UCSC genome browsera existujú aj ďalšie podobné webstránky, napríklad

- Ensembl (http://www.ensembl.org/index.html)
- Flybase (http://flybase.org/)
- SGD (Saccharomyces Genome database) (http://www.yeastgenome.org/)
- Iné špecializované databázy modelových organizmov (http://gmod.org/wiki/GMOD_Users)
- NCBI Mapview (http://www.ncbi.nlm.nih.gov/projects/mapview/) a pod.

Naštudujte si niektorú z nich a zistite, aké poskytuje možnosti na ručnú prácu a v akej forme sa dajú exportovať dáta.

Cvičenia 3

Prednáška 3 • Odovzdávanie D.Ú.3 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=14)

Pri odpovedaní týchto otázok si skúste napísať poznámky o tom, čo ste počas práce na domácej úlohe museli o UCSC browseri zistiť z návodov alebo skúšaním. V učebni sa pokúste spraviť hlavne časti A, E, F a G, na ostatné časti potrebujete iba internetový prehliadač a ľahšie ich dorobíte doma.

- **A** V UCSC genome browseri http://genome.ucsc.edu/ si zvoľte ľudský genóm. Do okienka `position or search term` zadajte HOXA2. Vo výsledkoch hľadania zvoľte gén homeobox A2. Koľko má exónov? Na ktorom chromozóme a pozícii je? Pozor, je na opačnom vlákne, ako je to naznačené na obrázku?
- **B** V tracku UCSC Genes kliknite na gén, mali by ste sa dostať na stránku popisujúcu jeho rôzne vlastnosti. Čo ste sa dozvedeli o jeho funkcii?
- **C** Pomocou tracku SNPs (131) nájdite aspoň jeden single nucleotide polymorphism v tomto géne. Na akej je pozícii? Aké bázy sa vyskytujú v ľudskej populácii? Prečo sú niektoré SNPy zobrazené červenou?
- **D** Pomocou Placental Chain/Net zistite na akom chromozóme je homológ tohto génu v myši.
- **E** Pomocou nástroja Tables na modrej lište vyextrahujte proteínovú sekvenciu tohto génu:
  - Skôr než prejdete do časti tables, uistite sa, že máte na obrazovke zobrazený HOX2A gén (a žiaden iné gény)
  - V Tables skontrolujte, že máte vybranú tabuľku (table) knownGene
  - V časti region zvoľte možnosť position
  - V časti output format zvoľte sequence
  - Do okienka output file napíšte protein.fa
  - Stlačte get output a na ďalšej obrazovke zvoľte protein. Súbor protein.fa by sa vám mal uložiť do adresára Desktop v domovskom adresári na počítači, za ktorým sedíte.

- **F** Otvorte si nové okno s príkazovým riadkom a pomocou príkazu ls overte, že súbor naozaj máte. Môžete použiť aj príkaz cd, aby ste sa presunuli do adresára, v ktorom sa tento súbor nachádza. Pozrite sa do súboru protein.fa príkazom less a skopírujte ho do protokolu. Uveďte aj linuxové príkazy, ktoré ste použili.
  - Ak robíte domácu úlohu na Windowsovom počítači, použite napr. notepad na prezretie súboru.
- **G** Súbor protein.fa skopírujte na server vyuka.compbio.fmph.uniba.sk pomocou príkazu scp: `scp protein.fa meno@vyuka.compbio.fmph.uniba.sk:` (slovo meno nahraďte vašim prihlasovacím menom, pred protein.fa pridajte cestu, ak nie ste v správnom adresári)
  - Ak chcete skontrolovať, že sa vám to podarilo, prihláste sa na server pomocou ssh a použite príkaz ls.
  - Ak robíte domácu úlohu na Windowsovom počítači, použite namiesto scp program winscp
- **H, nepovinná bonusová časť** Vráťte sa k HOX2A génu v UCSC browseri a skúste pomocou ďalších trackov zistiť o ňom alebo okolitej sekvencii ešte niečo zaujímavé.

Prednáška 4

Na tejto prednáške sa naučíme ďalšie príkazy na prácu so súbormi v Linuxe a vyskúšame si to na tvorbe jednoduchej webstránky.

Opakovanie

- mená adresárov a súborov zadávame pomocou absolútnej cesty (napr. `/projects/data-ppb/events`) alebo relatívnej cesty (napr. `../events`)
- na server sa prihlásime príkazom `ssh -XC meno@vyuka.compbio.fmph.uniba.sk`
- zoznam súborov v adresári si pozrieme príkazom `ls meno_adresara`
- obsah textového súboru si môžete pozrieť príkazom `less meno_suboru`
- do iného adresára sa presunieme príkazom `cd meno_adresara`
- súbor skopírujeme pomocou príkazu `cp stare_meno nove_meno`
- kopírovanie zo servera alebo na server `scp meno@vyuka.compbio.fmph.uniba.sk:meno_suboru .` alebo `scp README meno@vyuka.compbio.fmph.uniba.sk:meno_suboru`
- užitočné pomôcky na príkazovom riadku: kláves Tab na dopĺňanie mien, šípky hore a dole na listovanie históriou zadaných príkazov
- dávajte pozor, čo spúšťate, neexistuje undo
- vždy sú tu aj grafické programy ako dolphin alebo nautilus

Vyrábanie a mazanie adresárov (mkdir, rmdir)

Príkaz mkdir (make directories)

- `mkdir meno_adresara`
- vytvorí nový adresár s daným menom

Príkaz rmdir (remove empty directories)

- `rmdir meno_adresara`
- vymaže daný adresár, ale iba ak je prázdny

Príklad:

```
# Vytvoríme v aktuálnom adresári podadresár s názvom prvy:
mkdir prvy
# Bez zmeny aktuálneho adresára vytvoríme v adresári prvy jeho podadresár dalsi:
mkdir prvy/dalsi

# Bez zmeny aktuálneho adresára vymažeme z adresára druhy jeho podadresár dalsi:
rmdir prvy/dalsi
# Vymažeme v aktuálnom adresári podadresár s názvom prvy:
rmdir prvy
```

Mazanie a presúvanie súborov (rm, mv)

Príkaz rm (remove files)

- `rm subor`
- Vymaže zadanadaný súbor
- Napr. `rm events/README`
- Používať veľmi opatrne, nie je undo!

Príkaz mv (move/rename files)

- Podobne ako cp, ale zmaže súbor z pôvodného miesta a presunie ho na nové
- `mv odkial kam`
- Dá sa použiť na presúvanie a premenovávanie súborov aj adresárov

Práca s veľa súbormi naraz

- Mnohé príkazy (ls, cp, mv, rm, ...) vedia pracovať aj s viacerými súbormi
  - súbory vymenujeme po jednom alebo použijeme hviezdičky

```
# Skopírujeme dva súbory z aktuálneho adresára do domovského adresára
cp Fri_12_Mar_2010 Fri_19_Feb_2010 /home/hrasko37/

# Presunieme všetky súbory, ktorých meno začína na Fri do domovského adresára
mv Fri* /home/meno/

# Zmažeme všetky súbory s príponou .txt v aktuálnom adresári (OPATRNE)
rm *.txt
```

- Mazať a kopírovať môžeme aj celé adresáre so všetkými súbormi a podadresármi, čo v nich sú
  - Použijeme na to nastavenie -r, ktoré sa píše za meno príkazu (cp, ls), oddelené medzerou
  - Opatrne!

```
rm -r /projects/data-ppb/events
cp -r /projects/data-ppb/events /home/hrasko37/

#aj scp vie kopírovať celé adresáre:
scp -r hrasko37@vyuka.compbio.fmph.uniba.sk:/projects/data-ppb/events .
```

Editovanie textových súborov (gedit)

- `gedit subor` otvorí grafický editor gedit, podobný na Notepad z Windows
- Klasické textové linuxové editory vi, emacs sa ťažšie ovládajú
- Ak potrebujete jednoduchý textový editor, skúste nano

Práva súborov (ls -l, chmod, chgrp)

Určenie, **kto** môže čo s daným súborom **robiť**.

Tri kategórie užívateľov:

- **u**ser - človek, ktorý súbor vytvoril
- **g**roup - skupina ľudí, ktorí napríklad pracujú na spoločnom projekte s user-om
- **o**thers - ostatní užívatelia, ktorí nie sú v danej skupine

Čo môže/nemôže užívateľ **robiť**:

- **r**ead - čítať obsah súboru, pozerať zoznam súborov v adresári
- **w**rite - prepisovať súbor
- e**x**ecute - spúšťať program, vstupovať do adresára

Zisťovanie nastavených práv (ls -l)

- `ls -l`
- vypíše kopu údajov, okrem iného v prvých stĺpcoch aj práva:
  - za sebou user, group, others
  - každý má zápis v troch stĺpcoch r,w,x; ak tam je písmenko, daná činnosť je povolená, ak pomlčka, nie je
- druhý a tretí stĺpec obsahujú meno vlastníka a skupiny, ktorým súbor patrí.

Príklad: súbory v adresári /projects/data-ppb/events/2010 môže čítať a písať užívateľ bbrejova a čítať užívatelia v skupine ppb (čo sú študenti v tomto predmete):

```
bbrejova@vyuka:/projects/data-ppb/events/2010$ ls -l
total 36
-rw-r----- 1 bbrejova ppb 199 2010-02-21 18:45 Fri_12_Mar_2010
-rw-r----- 1 bbrejova ppb 240 2010-02-21 18:45 Fri_19_Feb_2010
-rw-r----- 1 bbrejova ppb  67 2010-02-21 18:45 Mon_1_Mar_2010
-rw-r----- 1 bbrejova ppb 105 2010-02-21 18:45 Mon_25_Jan_2010
-rw-r----- 1 bbrejova ppb 146 2010-02-21 18:45 Thu_11_Feb_2010
-rw-r----- 1 bbrejova ppb 159 2010-02-21 18:45 Tue_12_Jan_2010
-rw-r----- 1 bbrejova ppb  92 2010-02-21 18:45 Tue_16_Feb_2010
-rw-r----- 1 bbrejova ppb 202 2010-02-21 18:45 Tue_23_Feb_2010
-rw-r----- 1 bbrejova ppb  76 2010-02-21 18:45 Wed_28_Apr_2010
```

Nastavovanie práv (chmod)

- `chmod ktoZnamienkoPrava meno_suboru`
  - *kto* je buď user (**u**) alebo group (**g**) alebo others (**o**) alebo all (**a**)
  - *Znamienko* je buď pridať (**+**) alebo odobrať (**-**)
  - *Prava* je read (**r**), write (**w**) alebo execute (**x**)

Príklad:

```
mkdir pokus
ls -l
  drwxrws--x 2 bbrejova ppb 4096 2010-02-28 10:55 pokus

# zakážeme ostatným prístup do adresára
chmod o-x pokus
ls -l
  drwxrws--- 2 bbrejova ppb 4096 2010-02-28 10:55 pokus

# zakážeme všetkým (aj sebe) meniť adresár
```

```
chmod a-w pokus
ls -l
  dr-xr-s--- 2 bbrejova ppb 4096 2010-02-28 10:55 pokus
mkdir pokus/pokus2
  mkdir: cannot create directory `pokus/pokus2': Permission denied

# dovolíme sebe meniť adresár
chmod u+w pokus
ls -l
  drwxr-s--- 2 bbrejova ppb 4096 2010-02-28 10:55 pokus
mkdir pokus/pokus2

# písmenka môžeme kombinovať: povolíme čítanie a písane sebe aj skupine:
chmod gu+rw pokus

# zmenenie práv všetkým súborov a podadresárom v danom adresári
chmod -R g+w pokus
```

Zmena skupiny chgrp

- `chgrp pokus ppb`
  - Zmení skupinu adresára pokus na ppb
- `chgrp pokus bbrejova`
  - Zmení na skupinu bbrejova (v ktorej je len užívateľ bbrejova)

Nastavenia príkazov, man

- Ako sme videli, niektorým príkazom sa dajú meniť nastavenia pomocou pomlčiek
- Napr. ls vypíše zoznam súborov, ls -l vypíše aj ďalšie informácie
- ls -a ukáže aj skryté súbory, ktorých meno začína na .
- nastavenia sa dajú aj kombinovať, napr. `ls -l -a` alebo `ls -la` vypíše aj skryté súbory a navyše aj všetky informácie o každom

Ďalšie príklady nastavení

- chmod mení práva danému súboru, chmod -R aj všetkým súborom v adresári
- cp -r alebo rm -r mažú/kopírujú celé adresáre
- nastavenie -i pri mv, cp a rm sa pýta skôr než niečo zmaže alebo prepíše iným súborom

Kde hľadať informácie o nastaveniach

- Prehľad nastavení (options) k príkazu sa dá nájsť príkazom `man` (skratka od manual)
- Napr `man ls`
  - Tiež ak niečo neviete zistiť v man-e, vždy je tu google

Tip na projekt

- Na dnešnom cvičení budete robiť malú zmenu v jednoduchej webstránke. Na projekt sa môžete naučiť viac o HTML a vytvoriť webstránku, pokiaľ možno s obsahom týkajúcim sa biológie alebo bioinformatiky (môže napríklad vysvetľovať podrobnejšie niektoré cvičenie, ktoré budeme tento semester robiť).

Ďalšie materiály

- Tutoriál Linuxu (http://www.fsid.cvut.cz/cz/U201/linux.html) a ešte jeden (http://tldp.org/LDP/gs/node5.html)

- Viac o hviezdičkovej notácii (http://www.tuxfiles.org/linuxhelp/wildcards.html)

K jazyku HTML na tvorbu webstránok:

- Tutoriál HTML (http://www.w3schools.com/html/html_intro.asp) a ďalší (http://www.davesite.com/webstation/html/) a ešte jeden (http://www.mcli.dist.maricopa.edu/tut/)
- Jazyk JavaScript na vytváranie dynamických stránok príklady, napr. časť Popup Boxes (http://www.w3schools.com/js/js_examples.asp) ,kvíz (http://www.mcli.dist.maricopa.edu/show/interact/js_ex2.html)

Cvičenia 4

Odovzdávanie D.Ú.4 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=16) • Prednáška 4

Poznámka: Pri známkovaní domácej úlohy sa okrem protokolu bude kontrolovať, aj či máte správne súbory s webstránkou vo vašom adresári.

- **A** Pomocou `man ls` zistite, čo robí nastavenie -h. Do protokolu skopírujte príslušný kúsok manuálu.
  - Skúste si toto nastavenie spolu s nastavením -l na adresári `/projects/data-ppb/ukazka/`
  - Do protokolu skopírujte príkaz, ktorý ste použili, aj jeho výsledok. Ktorý súbor je najväčší?

- **B** Pozrite si webstránku http://vyuka.compbio.fmph.uniba.sk/lsb09-web/drosophila/
  - V tejto domácej úlohe postupne vytvoríte **kópiu tejto stránky a zmodifikujete ju**. Tým si **precvičíte prácu so súbormi a právami**.
  - Súbory k tejto webstránke nájdete na serveri v adresári /var/www/lsb09-web/drosophila
  - Pomocou príkazu ls -l (s vhodným adresárom ako argumentom) zistite, aké súbory sú v tomto adresári. Do protokolu dajte príkaz, jeho výstup a slovne popíšte, aké majú súbory práva, kto je ich user a group a ktorý deň boli naposledny menené.

- **C** Vytvorte adresár public_html v svojom domovskom adresári a spravte ho čitateľný a spustiteľný pre každého.
  - Poznačte si príkazy, ktoré ste použili a skontrolujte výsledok pomocou ls -l
  - Tento adresár by sa teraz mal objaviť na webe ako http://vyuka.compbio.fmph.uniba.sk/~vasemeno/ Skontrolujte vo firefoxe.

- **D** Skopírujte adresár /var/www/lsb09-web/drosophila ako podadresár svojho public_html
  - Príkazom cp -r alebo v nejakom grafickom programe ako napr. dolphin
  - Nastavte práva príkazom chmod, aby adresár drosophila bol čitateľný a spustiteľný pre každého a súbory v ňom súbory boli čitateľné pre každého. Skontrolujte pomocou ls -l
  - Do protokolu napíšte príkazy, ktoré ste použili a výsledok ls -l
  - Skontrolujte vo firefoxe, že teraz je adresár na webe ako http://vyuka.compbio.fmph.uniba.sk/~vasemeno/drosophila/

- **E** V adresári sú tri dátové súbory a súbor index.html, ktorý obsahuje linky na dva z

dátových súborov.
    - Otvorte súbor index.html v editore gedit a zmeňte ho tak, aby obsahoval linky na všetky tri súbory, pričom postupujete podľa vzoru v už existujúcich linkách (aj keď neviete HTML). Do protokolu skopírujte text, ktorý ste pridali do index.html.
    - Skontrolujte výsledok vo firefoxe.

- **Poznámky:**
    - Takýmito jednoduchými webstránkami môžete napríklad **zdieľať dáta** s kolegami, najmä väčšie súbory, ktoré sa zle posielajú e-mailom, alebo ak chcete k jednotlivým súborom pridať vysvetľujúci komentár.
    - Súbor index.html je napísaný v jazyku **HTML**.
    - Jednoduché stránky spravíte modifikáciou tohto príkladu
    - Word a iné grafické editory vedia exportovať do html
    - Prípadne sa HTML môžete naučiť podľa tutoriálov na webe

- **F (nepovinná)** Ak zdieľame citlivejšie dáta a nechceme, aby si ich hocikto na svete mohol stiahnuť, môžeme stránku **zaheslovať**:
    - V svojom domovskom adresári spustite príkaz `htpasswd -c web_passwords Mrkvicka`
    - Počítač si vypýta heslo pre užívateľa Mrkvicku a dajte mu heslo Hrasok (musíte ho ako obvykle zadať dvakrát).
    - Aký súbor príkaz vytvoril? Čo je v ňom? Spravte ho čitateľný pre každého.
    - V public_html/drosophila editorom vytvorte súbor .htaccess s týmto obsahom (vasemeno nahraďte vaším prihlasovacím menom):

```
AuthName "Secret data"
AuthType Basic
AuthUserFile /home/vasemeno/web_passwords
require valid-user
```

-
    - Nastavte mu práva na čítanie pre všetkých.
    - Vo firefoxe skontrolujte, že stránka vyžaduje meno a heslo a skúste ho zadať.
    - Všimnite si, že príkaz ls nezobrazí súbor .htaccess (iba ak použijete ls -a)
    - Nezabudnite do protokolu napísať všetky príkazy, ktoré ste použili, výstup programov, kde treba a prípadné ďalšie poznámky.

Prednáška 5

Úvod do programovania v jazyku Perl

Prvý program: hello world

```
#! /usr/bin/perl -w
use strict;
print "Hello world!\n";
```

- Prvý program nerobí nič iné, iba vypíše text Hello world!
- Prvý riadok vraví, že ideme používať perl.
- Druhý zapne lepšie odhaľovanie chýb.
- Tretí riadok vypisuje text, ktorý musí byť v úvodzovkách.
- \n znamená koniec riadku.
- Program si uložíme do súboru, napr. hello.pl

- Nastavíme mu práva na spustenie (`chmod u+x hello.pl`)
- Spustíme ho príkazom `./hello.pl`

Druhý program: počítanie riadkov

- Napíšeme program, ktorý bude počítať počet riadkov v súbore.

```
#! /usr/bin/perl -w
use strict;

my $pocet = 0;
while (my $riadok = <>) {
    $pocet = $pocet + 1;
}
print $pocet, "\n";
```

- $pocet je meno **premennej**: chlieviku v pamäti, kam si budeme ukladať nejaké hodnoty
- $pocet bude obsahovať počet riadkov, ktoré sme doteraz videli
- Na začiatku sme nevideli žiadne riadky, $pocet nastavíme na 0, slovíčkom `my` vytvoríme novú premennú
- Príkaz `while (my $riadok = <>)` je **cyklus**, ktorý ide cez všetky riadky na vstupe
  - <> je príkaz na načítanie jedného riadku zo súboru zadaného na príkazovom riadku alebo jedného riadku od užívateľa
  - premenná $riadok bude obsahovať tento riadok
- Pre každý riadok sa spustí všetko medzi zloženými zátvorkami {, }
  - V našom prípade iba príkaz `$pocet = $pocet + 1;`
  - Ten zvýši hodnotu uloženú v $pocet o 1
- Na konci vypíšeme premennú $pocet a koniec riadku "\n"

Tretí program: vypísanie dlhých riadkov

Tento program vypíše iba riadky s aspoň desať znakmi (vrátane znaku pre koniec riadku).

```
#!/usr/bin/perl -w
use strict;

while (my $riadok=<>) {
    if(length($riadok) >= 10) {
        print $riadok;
    }
}
```

- V premennej $riadok máme uložený práve načítaný riadok
- length($riadok) spočíta jeho dĺžku
- >= znamená "väčšie alebo rovné"
- length($riadok) >= 10 je pravda ak dĺžka riadku je spoň 10
- príkaz if testuje, či je podmienka splnená a ak áno, vykoná príkazy medzi { a } za ním

Štvrtý program: nájdenie najdlhšieho riadku

```
#!/usr/bin/perl -w
```

```
use strict;

my $max_riadok = "";
while (my $riadok=<>) {
    if(length($riadok) > length($max_riadok)) {
        $max_riadok = ?? (doplnte)
    }
}
print "Najdlhsi riadok ma dlzku: ", length($max_riadok), "\n";
print $max_riadok;
```

- V premennej $max_riadok máme najldhší riadok, aký sme doteraz našli
- Na začiatok si tam dáme prázdny riadok dĺžky 0
- Vždy, keď načítame riadok, porovnáme jeho dlku s dĺžkou uloženého riadku
- Na konci vypíšeme riadok, ktorý nám ostal v premennej $max_riadok a jeho dĺžku

Ďalšie informácie o Perle

- Viacero man stránok o Perle:
  - **man perlintro** úvod do Perlu
  - **man perlfunc** zoznam štandardných funkcií v Perle (napr. split, join, length a mnoho ďalších)
  - **perldoc -f split** vám povie viac o funkcii split, podobne pre iné funkcie
  - **perldoc -q sort** vám ukáže odpovede na často kladené otázky (FAQ)§
  - **man perlretut** a **man perlre** o regulárnych výrazoch, čo je veľmi šikovný nástroj na spracovanie textu
  - **man perl** zoznam ďalších man stránok o Perle
- Rôzne tutoriály na webe, napr. tu (http://www.perl.com/pub/a/2000/10/begperl1.html)
- Knihy
  - Simon Cozens: Beginning Perl [2] (http://www.perl.org/books/beginning-perl/) zadarmo na internete
  - Larry Wall a spol: Programming Perl [3] (http://oreilly.com/catalog/9780596000271/) klasika, Camel book
- **Bioperl** [4] (http://www.bioperl.org/wiki/Main_Page) veľká knižnica užitočných funkcií pre bioinformatiku
- Perl pre Windows: http://strawberryperl.com/

Cvičenia 5

Prednáška 5 • Odovzdávanie D.Ú.5 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=17)

Poznámka: Pri známkovaní domácej úlohy sa okrem protokolu môže kontrolovať, aj či máte súbory s programami vo vašom adresári na serveri.

Časť A: Program hello world

```
#! /usr/bin/perl -w
use strict;
print "Hello world!\n";
```

- Prihláste sa na server, vytvorte si adresár DU5 vo svojom domovskom adresári

(mkdir), presuňte sa do neho (cd) a zvyšok úlohy robte v ňom.
- Spustite editor príkazom `gedit hello.pl &`, nakopírujte si tam program, uložte ho.
- Vďaka symbolu & zostane gedit bežať v pozadí v inom okne a vy môžete stále používať príkazový riadok a nemusíte stále zatvárať a otvárať editor.
- Nastavte programu práva na spustenie (`chmod u+x hello.pl`)
- Spustite ho ako `./hello.pl`
- Zmodifikujte program tak, aby vypísal dva riadky:
  - `Hello world!`
  - `Good morning sunshine!`
- Do protokolu si skopírujte príkazy, ktoré ste použili a váš zmenený program.

Časť B: počítanie riadkov

```
#! /usr/bin/perl -w
use strict;

my $pocet = 0;
while (my $riadok = <>) {
    $pocet = $pocet + 1;
}
print $pocet, "\n";
```

- Uložte tento program do súboru pocitaj.pl, nastavte mu práva a spustite ho (podobne ako v časti A, nemusíte už uvádzať príkazy do protokolu).
- Program čaká, čo napíšete. Napíšte niekoľko riadkov a stlačte Ctrl+D. Skopírujte si, čo ste písali vy a program.
- Do súboru vstup.txt uložte editorom nasledujúce dáta:

```
123456789
0123456789
a
ab
abc
```

- Spustite `./pocitaj.pl vstup.txt` a skopírujte si odpoveď
- Súbor /var/www/lsb09-web/drosophila/droMelGenes.gp (ktorý máte skopírovaný aj vo svojej webstránke) obsahuje na každom riadku údaje o jednom géne. Koľko je v ňom génov?

Časť C: Modifikácia programu pocitaj.pl

- Zmodifikujte program pocitaj.pl tak, aby okrem počtu riadkov vypísal aj počet znakov v súbore. Ako na to:
  - Potrebujeme vytvoriť a priebežne udržiavať dve premenné: $pocet ktorá obsahuje počet riadkov a napr. $znaky, ktorá obsahuje počet znakov
  - V premennej $riadok máme vždy všetky znaky aktuálneho riadku, length($riadok) spočíta ich počet
  - Výsledný program vyskúšajte na súbore vstup.txt
  - Hotový program si skopírujte do protokolu.

Časť D: Preskočenie prvého riadku

- Napíšte program preskoc.pl, ktorý vypíše celý vstupný súbor okrem prvých dvoch

riadkov. T.j. na vstup.txt vypíše

```
a
ab
abc
```

- Použite premennú $pocet, podobne ako v pocitaj.pl, v ktorej máme, koľko riadkov sme doteraz videli.
- V treťom programe na prednáške vypisujeme iba niektoré riadky pomocou podmienky if. Ako podmienku zmeniť, aby fungovala?
- Výsledný program vyskúšajte na súbore vstup.txt.
- Hotový program si skopírujte do protokolu.

Časť E (nepovinná): Číslovanie riadkov

- Napíšte program, ktorý vypíše vstupný súbor a na začiatku každého riadku pridá číslo riadku
  - Program sa bude podobať na pocitaj.pl
  - Výsledný program a príklad jeho výstupu si zapíšte do protokolu

Časť F (nepovinná): Kde je najdlhší riadok?

- Modifikujte program na hľadanie najdlhšieho riadku z prednášky tak, aby vypísal aj číslo riadku, na ktorom sa tento najdlhší riadok v súbore nachádza (napr. pre vstup.txt by to mal byť riadok č. 2).
  - Výsledný program a príklad jeho výstupu si zapíšte do protokolu

Prednáška 6

Domáca úloha 6 (3%) · Odovzdávanie D.Ú.6 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=18)

Vstupné dáta: opakovania

- Na stránke http://genome.ucsc.edu/ zvolíme genóm drozofily, potom v časti Tables zvolíme group: variation and repeats, track: ReapatMasker, region: position chr2L, output format: all fields from the selected table a output file: repeats.txt
- Dostávame súbor, ktorý nájdete na serveri pod menom /projects/data-ppb/du6/repeats.txt
- Na každom riadku obsahuje údaje o jednom opakovaní na zvolenom chromozóme. Výnimkou je prvý riadok s menami stĺpcov. Stĺpce sú oddelené tabulátorom. Tu sú prvé dva riadky:

```
#bin    swScore milliDiv        milliDel        milliIns        genoName        genoStart       genoEnd genoLeft
585     778     167     7       20      chr2L   1       154     -23011390       +HETRP_DM       Satellite
```

- Počet riadkov je 12876 (vedeli by sme spočítať programom s minulej prednášky)
- Budeme používať súbor /projects/data-ppb/du6/repeats2.txt v ktorom je vynechaná hlavička (podobný program sme robili na DÚ 5)
- Budeme používať aj súbor /projects/data-ppb/du6/repeats3.txt v ktorom je len 6

opakovaní:

```
596     185     94      0       30      chr2L   1544990 1545023 -21466521       +       (AAATG)n        Simple_r
620     24      0       0       0       chr2L   4654439 4654463 -18357081       +       AT_rich Low_complexity
680     291     270     0       20      chr2L   12540437        12540539        -10471005       +       (CA)n
712     217     0       0       57      chr2L   16663977        16664012        -6347532        +       (CTTTG)n
94      15107   268     20      31      chr2L   22540111        22545932        -465612 +       Gypsy2_I
758     9763    105     124     6       chr2L   22714176        22716800        -294744 +       NINJA_I LTR
```

Polia

- Na minulej prednáške sme používali premenné, v ktorých sme si vedeli uchovať 1 hodnotu, napr. $riadok, $pocet
- Pole je postupnosť hodnôt (niečo ako jeden stĺpec Excelovej tabuľky)
- Vytvorenie poľa: my @tabulka;
- Prvé políčko poľa: $tabulka[0]
- Druhé políčko poľa: $tabulka[1]
- Počet políčok v poli: scalar(@tabulka)

Dĺžka opakovania

- Chceme pre každé opakovanie vypísať iba dve hodnoty: jeho typ a dĺžku
- Riadok si rozbijeme na jednotlivé stĺpce do poľa @stlpce

```
#!/usr/bin/perl -w
use strict;

while (my $riadok=<>) {

    #odstran koniec riadku
    chomp $riadok;

    #rozbi riadok na stlpce oddelene tabulatormi
    my @stlpce = split "\t", $riadok;

    #spocitaj dlzku opakovania
    my $dlzka = $stlpce[7]-$stlpce[6];

    print $stlpce[11], "\t", $dlzka, "\n";
}
```

Spustíme na krátkom súbore a dostávame:

```
meno@vyuka:/projects/data-ppb/du6$ ./dlzka.pl repeats3.txt
Simple_repeat   33
Low_complexity  24
Simple_repeat   102
Simple_repeat   35
LTR     5821
LTR     2624
```

Úprava programov

Je zvykom dodržiavať pri písaní programov určité pravidlá, aby boli programy prehľadné a dobre čitateľné:

- Jednotlivé príkazy dávame na zvláštne riadky a logické kusy programu môžeme oddeliť voľným riadkom

- Príkazy medzi { a } v cykle alebo podmienke odsadzujeme ďalej od začiatku riadku a } dáme pod podmienku alebo cyklus, kde sa { začína. Tak dobre vidno, odkiaľ pokiaľ telo cyklu alebo podmienky ide.
- Riadky začínajúce # sú komentáre a počítač ich ignoruje, my si tam môžeme písať vysvetľujúce poznámky
- Mená premenných by mali vyjadrovať ich význam.

Priemer

Ďalší program spočíta priemernú dĺžku opakovania. Potrebuje si počas čítania súboru udržovať dve premenné: počet opakovaní a súčet dĺžok opakovaní. Po prečítaní celého súboru ich vydelíme.

```
#!/usr/bin/perl -w
use strict;

#sucet dlzok opakovani
my $sucet = 0;
#pocet opakovani v subore
my $pocet = 0;

while (my $riadok=<>) {

    #odstran koniec riadku
    chomp $riadok;

    #rozbi riadok na stlpce oddelene tabulatormi
    my @stlpce = split "\t", $riadok;

    #spocitaj dlzku opakovania
    my $dlzka = $stlpce[7]-$stlpce[6];

    #zvys pocet a sucet dlzok
    $sucet = $sucet + $dlzka;
    $pocet = $pocet + 1;
}

#spocitaj priemer
my $priemer = $sucet/$pocet;

#vypis vysledky
print "Pocet opakovani: ", $pocet, "\n";
print "Sucet dlzok opakovani: ", $sucet, "\n";
print "Priemerna dlzka opakovania: ",  $priemer, "\n";
```

Takéto dostávame výsledky:

```
meno@vyuka:/projects/data-ppb/du6$ ./priemer.pl repeats3.txt
Pocet opakovani: 6
Sucet dlzok opakovani: 8639
Priemerna dlzka opakovania: 1439.83333333333

meno@vyuka:/projects/data-ppb/du6$ ./priemer.pl repeats2.txt
Pocet opakovani: 12875
Sucet dlzok opakovani: 1971749
Priemerna dlzka opakovania: 153.145553398058
```

Môže sa nám stať, že program omylom spustíme na súbore, ktorý nemá správny formát. Toto dostaneme:

```
meno@vyuka:/projects/data-ppb/du6$ ./priemer.pl priemer.pl
Use of uninitialized value in subtraction (-) at ./priemer.pl line 18, <> line 1.
Use of uninitialized value in subtraction (-) at ./priemer.pl line 18, <> line 1.
Use of uninitialized value in subtraction (-) at ./priemer.pl line 18, <> line 2.
```

```
...
Use of uninitialized value in subtraction (-) at ./priemer.pl line 18, <> line 28.
Use of uninitialized value in subtraction (-) at ./priemer.pl line 18, <> line 28.
Pocet opakovani: 28
Sucet dlzok opakovani: 0
Priemerna dlzka opakovania: 0

meno@vyuka:/projects/data-ppb/du6$ ./priemer.pl prazdny.txt
Illegal division by zero at ./priemer.pl line 24.
```

Kontrola vstupu

- Vstup programu je preto dobre overovať, aby sme neverili naslepo výsledkom, ktoré sme dostali zo zlých dát.
- Na prednáške to pre prehľadnosť programov vždy nerobíme, ale v praxi vrele doporučujem!
- != znamená "nerovná sa"
- Príkaz die zastaví program a vypíše chybovú hlášku.

```
#!/usr/bin/perl -w
use strict;

#sucet dlzok opakovani
my $sucet = 0;
#pocet opakovani v subore
my $pocet = 0;

while (my $riadok=<>) {

    #odstran koniec riadku
    chomp $riadok;

    #rozbi riadok na stlpce oddelene tabulatormi
    my @stlpce = split "\t", $riadok;

    #skontroluj spravny pocet riadkov
    if(scalar(@stlpce) != 17) {
        die "Riadok so zlym poctom stlpcov:\n", $riadok;
    }

    #spocitaj dlzku opakovania
    my $dlzka = $stlpce[7]-$stlpce[6];

    #skontroluj, ci je dlzka kladne cislo
    if($dlzka<=0) {
        die "Dlzka opakovania musi byt kladna:\n", $riadok;
    }

    #zvys pocet a sucet dlzok
    $sucet = $sucet + $dlzka;
    $pocet = $pocet + 1;
}

#vypis pocet a sucet dlzok
print "Pocet opakovani: ", $pocet, "\n";
print "Sucet dlzok opakovani: ", $sucet, "\n";

#ak sme nacitali nejake opakovania, spocitaj priemer
if($pocet>0) {
    my $priemer = $sucet/$pocet;
    print "Priemerna dlzka opakovania: ",  $priemer, "\n";
}
#ak sme nenacitali ziadne opakovania, preskoc priemer
else {
    print "Priemer nemozno spocitat.\n";
}
```

A takéto dostávame výsledky:

```
meno@vyuka:/projects/data-ppb/du6$ ./priemer2.pl repeats2.txt
Pocet opakovani: 12875
Sucet dlzok opakovani: 1971749
Priemerna dlzka opakovania: 153.145553398058

meno@vyuka:/projects/data-ppb/du6$  ./priemer2.pl priemer.pl
Riadok so zlym poctom stlpcov:
#!/usr/bin/perl -w at ./priemer2.pl line 19, <> line 1.

meno@vyuka:/projects/data-ppb/du6$ ./priemer2.pl prazdny.txt
Pocet opakovani: 0
Sucet dlzok opakovani: 0
Priemer nemozno spocitat.
```

Filter

- Ešte jeden jednoduchý program: vypíšme iba opakovania typu Simple repeat.
- Znamienko eq zistí, či sú dva reťazce rovnaké.

```
#!/usr/bin/perl -w
use strict;

while (my $riadok=<>) {

    #odstran koniec riadku
    chomp $riadok;

    #rozbi riadok na stlpce oddelene tabulatormi
    my @stlpce = split "\t", $riadok;

    #vypis riadok iba ak je typu Simple_repeat
    if($stlpce[11] eq "Simple_repeat") {
        print $riadok, "\n";
    }
}
```

Obrázok

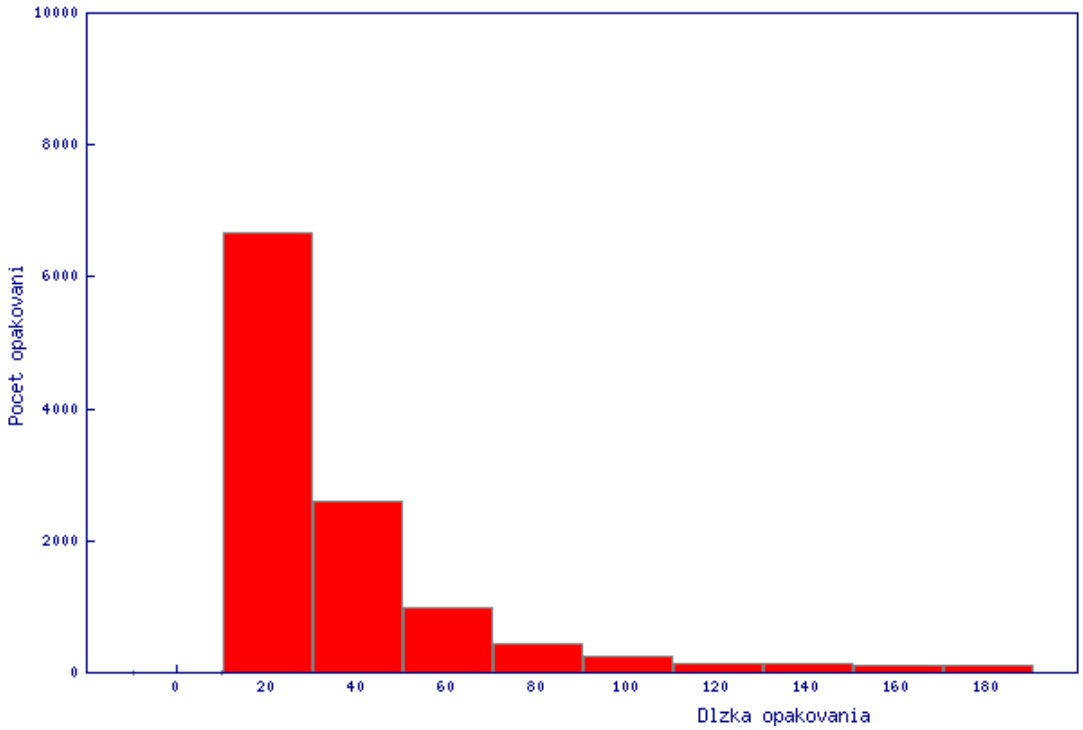
- A nakoniec jeden zložitý program, ktorý vykreslí histogram dĺžok opakovaní.
- Dĺžky rozdelíme do skupín po 20: od 0 do 19, od 20 do 39, od 40 do 59 atď
- Pre každú skupinu si v poli @stat pamätáme, koľko opakovaní z tejto skupiny sme videli
- POSIX::floor zaokrúhli číslo nadol
- Keď prečítame súbor, vytvoríme pole @y iba s prvými 10 skupinami (t.j. dĺžky menej ako 200)
- Pole @x bude mať pre každú skupinu jej minimálnu dĺžku, t.j. 0, 20, 40, ...
- Polia @x a @y vypíšeme na obrazovku a potom z nich zostavíme stĺpcový graf pomocou knižnice GD::Graph::bars a ten zapíšeme do súboru graf.png
- Graf si môžete pozrieť príkazom eog graf.png
- Príkaz for(my $skupina=0; $skupina<$pocet_skupin; $skupina++) prejde cez všetky hodnoty premennej $skupina od 0 po $pocet_skupin-1 a pre každú spraví príkazy medzi { }
- $premenna++ je to isté ako $premenna = $premenna + 1 (ale funguje to, aj ak $premenna ešte nemá žiadnu hodnotu)

```
#!/usr/bin/perl -w
use strict;

#pouzijeme dve kniznice
```

```
use GD::Graph::bars;
use POSIX;

#kolko hodnot do jedneho stlpca
my $velkost_skupiny = 20;
#kolko stplcov vykreslit do grafu
my $pocet_skupin = 10;

#nacitajme subor a pamatajme si pocet opakovani v kazdej skupine
my @stat;
while (my $riadok=<>) {

    #odstran koniec riadku
    chomp $riadok;

    #rozbi riadok na stlpce oddelene tabulatormi
    my @stlpce = split "\t", $riadok;

    #spocitaj dlzku opakovania
    my $dlzka = $stlpce[7]-$stlpce[6];

    #spocitaj cislo skupiny (stlpca grafu)
    my $skupina = POSIX::floor($dlzka/$velkost_skupiny);

    #zvys pocet opakovani v skupine
    $stat[$skupina]++;
}

#docitali sme subor
#spocitajme x-ovu a y-ovu suradnicu kazdeho stlpca
my @x;
my @y;
for(my $skupina=0; $skupina<$pocet_skupin; $skupina++) {
    #kolko je opakovani v skupine?
    my $pocet = $stat[$skupina];
    #musime sa vyrovnat s chybajucimi hodnotami
    if(!defined $pocet) {
        $pocet = 0;
    }
    #x-ova suradnica: znovu prenasov velkostou
    $x[$skupina] = $skupina*$velkost_skupiny;
    $y[$skupina] = $pocet;

    print "Pocet opakovani v skupine ", $x[$skupina],
    " je ", $y[$skupina], "\n";
}

#magia na vykreslenie obrazku do suboru
my $graf = GD::Graph::bars->new(600, 400);
$graf->set(
    "x_label" => "Dlzka opakovania",
    "y_label" => "Pocet opakovani",
    "transparent" => 0
);

my $obrazok = $graf->plot([\@x,\@y]);
open(IMG, '>graf.png') or die $!;
binmode IMG;
print IMG $obrazok->png;
close IMG;
```

Výsledok:

```
./graf.pl repeats2.txt
Pocet opakovani v skupine 0 je 7
Pocet opakovani v skupine 20 je 6678
Pocet opakovani v skupine 40 je 2595
Pocet opakovani v skupine 60 je 985
Pocet opakovani v skupine 80 je 447
Pocet opakovani v skupine 100 je 253
Pocet opakovani v skupine 120 je 139
Pocet opakovani v skupine 140 je 148
Pocet opakovani v skupine 160 je 114
Pocet opakovani v skupine 180 je 97
```



Ďalšie materiály

- Ďalšie informácie o Perle v prednáške 5
- Manuál ku knižnici GD::Graph (http://search.cpan.org/dist/GDGraph/Graph.pm)
- Zložitejšie nástroje na tvorbu grafov s širšími možnosťami: GNUplot (http://www.gnuplot.info/) , jgraph (http://www.cs.utk.edu/%7Eplank/plank/jgraph/jgraph.html)
- Štatistický systém R tiež poskytuje veľa možností na prácu s grafmi

Cvičenia 6

Prednáška 6 • Odovzdávanie D.Ú.6 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=18)

Nemusíte si už do protokolu písať všetky použité príkazy (mkdir, gedit, chmod a pod.), iba výsledné perlové programy a to čo vypíšu.

Časť A: Priemer

- Prihláste sa na server, vytvorte si adresár DU6 vo svojom domovskom adresári (mkdir), presuňte sa do neho (cd) a zvyšok úlohy robte v ňom.
- Príkazom cp skopírujte program /projects/data-ppb/du6/priemer.pl do svojho adresára DU6
- Upravte ho tak, aby rátal priemer iba opakovaní typu Simple_repeat (inšpirujte sa aj programom filter.pl z prednášky)
- Spustite váš nový program najprv na malom súbore: ./priemer.pl /projects/data-ppb/du6/repeats3.txt a potom na veľkom ./priemer.pl /projects/data-ppb/du6/repeats2.txt
- Výsledky aj zmenený program si zapíšte do protokolu. Porovnajte výsledok s hodnotami, ktoré sú v prednáške uvedené pre všetky typy opakovaní v súbore

repeats2.txt. Sú Simple repeats v priemere dlhšie alebo kratšie než ostatné typy opakovaní?

Časť B (nepovinná): Graf

- Príkazom cp skopírujte program /projects/data-ppb/du6/graf.pl do svojho adresára DU6
- Spustite ho na veľkom súbore pomocou ./graf.pl /projects/data-ppb/du6/repeats2.txt a výsledný obrázok si pozrite pomocou príkazu eog graf.png
- Upravte program tak, aby vypisoval v tabuľke nielen počet opakovaní v každej skupine, ale aj koľko je to percent zo všetkých opakovaní, napr. `Pocet opakovani v skupine 20 je 6678 (51%)`
  - Ako na to: pridajte si úplne na začiatok novú premennú, napr. $spolu, ktorá si bude pamätať celkový počet opakovaní, ktorý načítate vo while cykle.
  - Postupujte rovnako ako pri premennej $pocet v programe priemer.pl
  - Vo for-cykle premennou $spolu predelíte $pocet a výsledok uložte do novej premennej, napr. $percenta
  - Percentá môžete ešte prenásobiť 100 a zaokrúhliť na najbližšie celé číslo: $percenta = POSIX::floor($percenta*100 + 0.5);
  - Potom ich už len stačí pridať do príkazu print.
- Váš výsledný program spustite na /projects/data-ppb/du6/repeats2.txt a výsledok skopírujte do protokolu. Skopírujte tam aj zmenený program.

Časť C (nepovinná): Farby v grafe

- Pozrite si manuál ku knižnici GD::Graph (http://search.cpan.org/dist/GDGraph/Graph.pm) a skúste zistiť, ako zmeniť farbu grafu (pomôcka: nastavenia, ktoré niečo majú s farbou majú v mene clr)
- Do protokolu uveďte úryvok z programu, kde meníte farbu + zopár riadkov okolo.
- V adresári DU6 vyrobte vašim programom nový obrázok so zmenenou farbou.

Prednáška 7

Domáca úloha 7 (5%) · Odovzdávanie D.Ú.7 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=19)

Hlavný cieľ dnešnej prednášky je precvičiť si základné prvky Perlu, ktoré sme už videli (premenné, cykly, podmienky). Najprv si spravíme ich prehľad, potom ich budeme precvičovať na "nezmyselných" programoch, ktoré vykresľujú jednoduché obrazce z hviezdičiek.

Prehľad základov Perlu

Premenné

- Vytvorenie: `my $meno = hodnota;`
- Zmena hodnoty (priradenie) `$meno = hodnota;`
  - Špeciálne zvýšiť o jedna: `$pocet = $pocet+1;` alebo `$pocet++;`
  - Zvýšiť o dva: `$pocet = $pocet+2;` alebo `$pocet+=2;`
- Hodnota môže byť nedefinovaná (undef), číslo alebo reťazec (text, po anglicky

string).
    - V premennej môžu byť aj iné veci pre pokročilých, napr. referencia na inú premennú.

Čísla a aritmetické výrazy

Príklady:

- 1, 1.5, -7, 1e5, 1e-5
- $x+6, $x*$y, $x/($a+$b)
- $x**3 (x na tretiu)
- exp($x) (e na x), log($x) prirodzený logaritmus z x, abs(x) absolútna hodnota, ...
- POSIX::floor($x) zaokrúhli nadol, treba dať use POSIX; do hornej časti programu

Práca s reťazcami

- "Ahoj", "\n" koniec riadku, "\t" tabulátor
- Spájanie pomocou operátoru . napr. $pozdrav = "Ahoj " . $meno . "!\n";
- length($text) je dĺžka textu v znakoch (vrátane medier, koncov riadkov a pod.)
- substr($text, $odkial, $dlzka) vráti časť textu
- uc($text) zmení všetky písmená na veľké
- chomp($riadok) odstráni koniec riadku na konci textu (ak tam nejaký je)
- @pole = split "\t", $text; rozdelí text na časti ak boli oddelené tabulátorom

Logické výrazy

- Ich hodnota je pravda (1) alebo nepravda (0 alebo undef)
- Porovnanie čísel $a<$b, $a<=$b, $a>$b, $a>=$b, $a==$b, $a!=$b
- Porovnanie reťazcov (abecedne): $a lt $b, $a le $b, $a gt $b, $a ge $b, $a eq $b, $ab ne $b (skratky od lower than, lower or equal, greater than, greater or equal, equal, not equal).
- Logické spojky && (a súčasne), || (alebo), ! negácia
- Príklad: length($text)>4 || uc($text) eq "AHOJ"

Cykly

- Cykly umožnujú nejakú časť programu opakovať veľa krát.

- while(logicky_vyraz) { prikazy }
- všetky príkazy v tele cyklu sa opakujú, kým je logický výraz splnený
- napríklad while (my $riadok=<>) { prikazy } opakuje príkazy pre každý riadok vstupu

- for(my $i=1; $i<=$n; $i++) { prikazy }
- opakuje prikazy $n krát, pričom premennú $i nastavuje na 1,2,...,$n
- to isté ako

```
my $i=1;
while($i<=$n) {
  prikazy
  $i++;
```

```
}
```

Podmienky

- Podmienky umožňujú vykonať určitú časť programu len za určitých okolností
- if(logicky_vyraz) { prikazy1 } else { prikazy2 }
- Ak je logický výraz splnený, vykonajú sa prikazy1, inak sa vykonajú prikazy2
- Časť else možno vynechať.

Iné

- Podmienky a cykly možno vkladať do seba.
- Polia.
- Príkaz print vypisuje na obrazovku, príkaz die skončí a vypíše chybu.
- Pozor na úpravu (odsadzovanie vnútri cyklov a podmienok)
- Komentáre začínajú #

Programy na vykresľovanie hviezdičiek

Program 1: plný obdĺžnik

Užívateľ zadá dve čísla m a n (na jednom riadku oddelené medzerou) a program vykreslí obdĺžnik z hviezdičiek s m riadkami a n stĺpcami.

```
#! /usr/bin/perl -w
use strict;

#nacitaj riadok
my $riadok = <>;
#rozdel ho na dve cisla m a n
my ($m, $n) = split " ", $riadok;

for(my $i=1; $i<=$m; $i++) {
   for(my $j=1; $j<=$n; $j++) {
       print "*";
   }
   print "\n";
}
```

Príklad vstupu a výstupu (prvý riadok je vstup):

```
meno@vyuka:~$ ./vykresli.pl
2 5
*****
*****
```

- Čo program vypíše pre vstup 0 3? Čo pre vstup 3 0?

Program 2: prázdny obdĺžnik

To isté, ako program 1, ale vypíše iba prázdny rámik, potrebuje m,n aspoň 2. Príklad vstupu a výstupu (prvý riadok je vstup):

```
meno@vyuka:~$ ./vykresli.pl
4 6
******
```

```
*    *
*    *
******
```

Postup:

- Vypíšeme prvý a posledný riadok zvlášť.
- Pre ostatné riadky vypíšeme *, potom n-2 medzier a zase *.

```
#! /usr/bin/perl -w
use strict;

#nacitaj riadok
my $riadok = <>;
#rozdel ho na dve cisla m a n
my ($m, $n) = split " ", $riadok;

if($m<2 || $n<2) {
    die "Prilis male m alebo n.\n";
}

#prvy riadok
for(my $j=1; $j<=$n; $j++) {
    print "*";
}
print "\n";

#riadky 2..m-1
for(my $i=2; $i<=$m-1; $i++) {
    print "*";
    for(my $j=1; $j<=$n-2; $j++) {
       print " ";
   }
   print "*\n";
}

#posledny riadok
for(my $j=1; $j<=$n; $j++) {
    print "*";
}
print "\n";
```

To isté ale inak: postupujeme ako v programe 1, ale pre každé i a j sa rozhodneme, či vypísať * alebo medzeru.

```
#! /usr/bin/perl -w
use strict;

#nacitaj riadok
my $riadok = <>;
#rozdel ho na dve cisla m a n
my ($m, $n) = split " ", $riadok;

if($m<2 || $n<2) {
    die "Prilis male m alebo n.\n";
}

for(my $i=1; $i<=$m; $i++) {
    for(my $j=1; $j<=$n; $j++) {
        if($i==1 || $i==$m || $j==1 || $j==$n) {
            print "*";
        }
        else {
            print " ";
        }
    }
    print "\n";
}
```

Program 3: horný trojuholník

Program načíta iba jedno číslo n a vypíše zo štvorca n x n iba pravý horný trojuholník takto:

```
meno@vyuka:~$ ./vykresli.pl
4
****
 ***
  **
   *
```

Program:

```
#! /usr/bin/perl -w
use strict;

#nacitaj riadok
my $n = <>;
chomp $n;

for(my $i=1; $i<=$n; $i++) {
   for(my $j=1; $j<$i; $j++) {
       print " ";
   }
   for(my $j=$i; $j<=$n; $j++) {
       print "*";
   }
   print "\n";
}
```

- Ako by sme spravili ľavý horný trojuholník?

Cvičenia 7

Prednáška 7 • Odovzdávanie D.Ú.7 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=19)

Do protokolu si napíšte perlové programy a to čo vypíšu, prípadne ďalšie poznámky ak chcete.

Časť A: Trojuholník

- Prihláste sa na server, vytvorte si adresár DU7 vo svojom domovskom adresári (mkdir), presuňte sa do neho (cd) a zvyšok úlohy robte v ňom.
- Napíšte program trojuholnik.pl, ktorý pre jedno vstupné číslo n vypíše trojuholník s n riadkami, pričom prvý riadok bude mať jednu hviezdičku, druhý tri, tretí päť atď., pričom všetky riadky budú vystredené. Tu je príklad:

```
4
   *
  ***
 *****
*******
```

- Do protokolu si poznačte výsledný program a príklad jeho vstupu a výstupu pre n=2 a n=5.

Časť B: Vianoce

Napíšte program vianoce.pl, ktorý načíta dve čísla m a n a vypíše vianočný stromček pozostávajúci z m trojuholníkov, každý s n riadkami, ako v časti A. Príklad:

```
2 3
  *
 ***
*****
  *
 ***
*****
```

- Do protokolu si poznačte výsledný program a príklad jeho vstupu pre nejaké hodnoty m a n.

Časť C (nepovinná): Biele Vianoce

Napíšte program biele.pl, ktorý robí to isté ako program v časti B, ale z vianočného stromčeka nakreslí iba obrys:

```
2 3
  *
 * *
*****
  *
 * *
*****
```

- Do protokolu si poznačte výsledný program a príklad jeho vstupu pre nejaké hodnoty m a n.

Prednáška 8

Domáca úloha 8 (6%) • Odovzdávanie D.Ú.8 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=20)

Prehľad

- Prednášky 2 a 4: základy práce v Linuxe, prihlasovanie, práca so súbormi, práva súborov, jednoduchá webstránka
- Prednáška 3: UCSC genome browser, vrátane získavania dát
- Prednášky 5, 6 a 7: Základy programovania v jazyku Perl, niektoré príklady s tabuľkou opakovaní v genóme z UCSC
- Dnes: spúšťanie programu BLAST, spracovanie textových súborov jednoduchými linuxovými príkazmi
- Nabudúce: spúšťanie ďalších programov

Hľadanie podobností, zarovnaní

- Jednou zo základných metód, ako sa dozvedieť niečo o novej sekvencii je porovnať ju s inými sekvenciami
- Podobnosť sekvencií často znamená homológiu (evolučnú príbuznosť) a podobnosť

funkcií
- Všímame si E-value: koľko rovnako silných podobností by sme očakávali čisto náhodou (čím menej, tým lepšie)
- Veľké množstvo programov s rôznym zameraním (DNA vs. proteíny, dlhé alebo kratšie sekvencie, rýchlosť, ...)

- Jeden z najznámejších programov **NCBI blast** [5] (http://blast.ncbi.nlm.nih.gov/) , `man blast`, resp. info na webe
- blastall: sekvencia vs. databáza, databázu pripravíme z fasta súboru programom formatdb
- bl2seq: porovná dve sekvencie bez formátovania
- rôzne módy, "program" podľa typu sekvencií [6] (http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=ProgSelectionGuide)
  - blastp compares an amino acid query sequence against a protein sequence database.
  - blastn compares a nucleotide query sequence against a nucleotide sequence database.
  - blastx compares the six-frame conceptual translation products of a nucleotide query sequence (both strands) against a protein sequence database. For bl2seq, the nucleotide should be the first sequence given.
  - psitblastn compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands) using a position specific matrix created by PSI-BLAST.
  - tblastn compares a protein query sequence against a nucleotide sequence database dynamically translated in all six reading frames (both strands). For bl2seq, the nucleotide should be the second sequence given.
  - tblastx compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database.

- Je veľa iných programov na hľadanie podobností, napr.**blat** [7] (http://www.kentinformatics.com/blatfaq.asp) , používaný v UCSC browseri
- Hodí sa len na takmer identické sekvencie (EST vs. genóm)
- Rýchly, pohodlný výstup

Fasta súbory

- Sekvencie (DNA, RNA, proteíny) vo **fasta formáte (http://www.ncbi.nlm.nih.gov/blast/fasta.shtml)** :
  - Prvý riadok: meno sekvencie za značkou >, ďalšie riadky samotná sekvencia.
  - Neznáme DNA bázy sa označujú N, neznáme amino kyseliny X (ďalšie IUPAC kódy [8] (http://www.dna.affrc.go.jp/misc/MPsrch/InfoIUPAC.html)).
  - Ďalšia sekvencia zase začína > a menom.
  - Príklad (http://vyuka.compbio.fmph.uniba.sk/lsb09-web/drosophila/drosophilas.fasta)
- Mnohé bioinformatické súbory sú textové, t.j. vieme si ich ľahko pozrieť napr. programom less
- Linux má šikovné programy na prácu s takýmito súbormi
- Takisto si ďalšie nástroje ľahko naprogramujeme v Perle

Príkazy na prácu s textovými súbormi

Príkaz wc (word count, počítanie slov a riadkov)

- Vypíše, koľko sa v súbore nachádza riadkov, slov a znakov.
- `wc [nastavenia] [subor]`
- Nastavenia:
  - -l vypíše iba počet riadkov
  - -w vypíše iba počet slov
  - -c vypíše iba počet bytov (znakov)

Príklad:

```
#spočítaj štatistiky pre súbor drosophilas.fasta
#- súbor má 7621 riadkov, 7621 slov, vyše 388 tisíc znakov
meno@vyuka:/var/www/lsb09-web/drosophila$ wc drosophilas.fasta
  7621   7621 388458 drosophilas.fasta

#vypíš iba počet riadkov
meno@vyuka:/var/www/lsb09-web/drosophila$ wc -l drosophilas.fasta
7621 drosophilas.fasta
```

Príkaz grep (hľadanie v súbore)

- Hľadá riadky, ktoré obsahujú nami zadaný výraz
- `grep [nastavenia] [hladany_vyraz] [subor]`

Príklad:

```
#nájdi všetky riadky s aspoň tromi N za sebou
meno@vyuka:/var/www/lsb09-web/drosophila$ grep NNN drosophilas.fasta
ATGGAAACGACGAATNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNTCTAATAACAAGAGTGAGTGCCTTGCTTTGACCAG
...

#nájdi všetky riadky s aspoň tromi n za sebou
# - nenájde nič, lebo neobsahuje malé n
meno@vyuka:/var/www/lsb09-web/drosophila$ grep nnn drosophilas.fasta

#nájdi všetky riadky s väčšítkom: meno sekvencie vo faste
# '''pozor!''' špeciálne znaky ako napr. >, <, $, a pod. treba dať do apostrofov
meno@vyuka:/var/www/lsb09-web/drosophila$ grep '>' drosophilas.fasta
>droMel
>droSec
>droYak
>droSim
```

- Grep má veľa nastavení, napr.
  - -i - case insensitive (malé a veľké písmené to isté)
  - -v - invert (iba riadky, čo neobsahujú hľadaný výraz)
  - -c - count (počet riadkov s nájdenými výrazmi)
- Namiesto obyčajných reťazcov môžeme hľadať aj zložitejšie regulárne výrazy
  - `grep '[0-9]' subor` hľadá riadky obsahujúce nejakú číslicu
  - `grep '^>' subor` hľadá riadky obsahujúce > na začiatku riadku
- Viac informácií `man grep` a wikipédia: grep (http://en.wikipedia.org/wiki/Grep) , wikipédia: regulárne výrazy, hlavne časť o POSIX (http://en.wikipedia.org /wiki/Regular_expression)

Príkaz sort

- Usporiada riadky súboru
- `sort [nastavenia] [subor]`
- Zaujímavé nastavenia:
  - -r reverse (od najväčšieho po najmenší)
  - -g usporiadava podľa čísel, nie podľa abecedy
  - -k [cislo] usporiadava podľa niektorého stĺpca

Príkaz uniq

- Z viacerých za sebou idúcich riadkov rovnakých vypíše iba jeden.
- S nastavením -c počíta

```
#prikaz cat jednoducho vypise subor:
meno@vyuka:~$ cat ukazka.txt
aa
aa
bb
aa
#a teraz vynechajme opakujuce sa riadky
meno@vyuka:~$ uniq ukazka.txt
aa
bb
aa
meno@vyuka:~$ uniq -c ukazka.txt
      2 aa
      1 bb
      1 aa
```

Príkazy head a tail

- Vypisujú iba začiatok alebo koniec súboru

```
#vypis prve dva suboru:
meno@vyuka:~$ head -n 2 ukazka.txt
aa
aa
#vypis posledne tri suboru:
meno@vyuka:~$ tail -n 3 ukazka.txt
aa
bb
aa
```

Jednoriadkové programy v Perl-e (one-liners)

- Perl sa dá sa jednoducho použiť na spracovanie súboru riadok po riadku:

```
meno@vyuka:~$ cat ukazka2.txt
1 2 3 4
a b c d

#vypisme prve a tretie slovo v kazdom riadku:
meno@vyuka:~$ perl -lane 'print $F[0]," ", $F[2]' ukazka2.txt
1 3
a c
```

- Použité nastavenia -l, -a, -n, -e čítajú súbor riadok po riadku a každý rozoberú na stĺpce podľa medzier a stlpce uložia do poľa @F, riadok do premennej $_
- Uvedené príkazy (`print $F[0]," ", $F[2]`) sa vykonajú pre každý riadok

- Viac informácií napr. pomocou `man perlrun`

Spájanie príkazov, presmerovanie vstupu a výstupu

- Ak chceme výstup programu uložiť do súboru, "presmerujeme" ho pomocou >

```
meno@vyuka:~$ perl -lane 'print $F[0]," ", $F[2]' ukazka2.txt > ukazka3.txt
meno@vyuka:~$ cat ukazka3.txt
1 3
a c
```

- Namiesto do súboru ho môžeme poslať ďalšiemu programu pomocou |
- Napr. ak je výstup moc dlhý presmerujeme ho do programu less
- Spojenie príkazov sort a uniq nechá z každého riadku iba jeden, aj keď nie sú pri sebe

```
#listovanie v dlhom vysledku
meno@vyuka:~$ grep NNN /var/www/lsb09-web/drosophila/drosophilas.fasta | less
#sort + uniq
meno@vyuka:~$ sort ukazka.txt | uniq
aa
bb
```

- Presmerovať môžeme aj vstup

```
#v tomto prípade oba príkazy robia to isté:
meno@vyuka:~$ uniq <ukazka.txt
aa
bb
aa
meno@vyuka:~$ uniq ukazka.txt
aa
bb
aa
```

- Zobáčiky hovoria, akým smerom sa informácie posúvajú
- Ešte raz pozor na špeciálne znaky v grepe a pod.:
  - Zle (prepíše súbor): `grep > /var/www/lsb09-web/drosophila/drosophilas.fasta`
  - Dobre (hľadá zobáčik) `grep '>' /var/www/lsb09-web/drosophila/drosophilas.fasta`

Rada: Ak spájate za seba viacero príkazov pomocou tyče |, po každom príkaze si najskôr vizuálne overte, že výsledok vyzerá správne pomocu príkazu less, a potom zapojte ďalší. Napr. ak chceme spočítať počet navzájom rôznych riadkov obsahujúcich písmená XXX:

```
grep XXX subor.txt | less
grep XXX subor.txt | sort | less
grep XXX subor.txt | sort | uniq | less
grep XXX subor.txt | sort | uniq | wc
```

Cvičenia 8

Prednáška 8 • Odovzdávanie D.Ú.8 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=20)

V tejto domácej úlohe budeme spúšťať program BLAST a tiež zisťovať základné charakteristiky textových súborov pomocou príkazov, ktoré ste sa naučili. Pre každé

cvičenie napíšte príkazy, ktoré ste použili a celkovú odpoveď, ak bola krátka, alebo zopár prvých riadkov, ak bola dlhá. Kontrolovať budem aj prítomnosť príslušných súborov vo vašom adresári na serveri vyuka.

Časť A: fasta súbor

- V svojom domovskom adresári si spravte adresár DU8 a nakopírujte si tam súbor /var/www/lsb09-web/drosophila/drosophilas.fasta
  - Tento súbor obsahuje niekoľko kúskov genomických sekvencií z rôznych druhov drosofily vo fasta formáte
  - Už sme ho používali aj na webstránke v DU4
- Zistite počet sekvencií v tomto fasta súbore. Použite spojenie príkazov grep a wc. Príkazy a ich výsledok si zapíšte do protokolu.

Časť B: extrahovanie sekvencie

- Zo súboru drosophilas.fasta chceme vyextrahovať sekvenciu pre droMel (*Drosophila melanogaster*) do súboru droMel.fasta a pre sekvenciu droYak (*Drosophila yakuba*) do súboru droYak.fasta
  - Použite na to program faOneRecord z balíčka UCSC Kent tools, napr takto: `faOneRecord drosophilas.fasta droMel > droMel.fasta`
  - Prvý argument (drosophilas.fasta) je meno vstupného súboru, druhý (droMel) je meno sekvencie, ktorú chcete vypísať, t.j. slovo za znakom > súbore. Výstup je presmerovaný do súboru pomocou >
  - Zopakujte pre droYak
- Pomocou príkazu `wc -lc *.fasta` overte, že máte tri fasta súbory a koľko riadkov a znakov majú
- Použité príkazy a ich výstup si zapíšte do protokolu.

Časť C: Hľadanie podobností, zarovnaní, blastn

- Porovnajte sekvencie v súboroch droMel.fasta a droYak.fasta pomocou programu BLAST.
  - `bl2seq -p blastn -i droMel.fasta -j droYak.fasta >blast.out`
  - Pozrite si výsledok príkazom less, napíšte prvých pár riadkov do protokolu
- Pridajte blastu nastavenie `-D 1` (t.j. bl2seq -D 1 -p ...), dostanete ľahšie spracovateľný výstup, skopírujte si z neho prvých zopár riadkov
  - Vo výstupe je každý riadok jedno zarovnanie, okrem horných troch (hlavička tabuľky)
- Zrátajte počet zarovnaní v tabuľke:
  - Príkaz `grep -v '#' blast.out` odfiltruje hlavičku tabuľky. Spojte ho s príkazom wc na počítanie riadkov.
- Ak pridáte blastu nastavenie `-e 0.01`, program nevypíše slabšie zarovnania (E-value musí byť najviac 0.01)
  - Koľko zarovnaní nájdete s týmto nastavením? (príkazy a výsledok do protokolu)
- Poznámka: prehľad ďalších nastavení nájdete pomocou `bl2seq | less` resp. `man blast` príp. na webe

Časť D: Štatistika výsledkov blastu

- Zo súboru blast.out, ktorý sme dostali s nastaveniami -E 0.01 a -D 1 chceme zrátať jednoduchú štatistiku
- Tretí stĺpec tabuľky obsahuje, koľko percent báz je v zarovnaní rovnakých. Aká je najmenšia a najväčšia hodnota v tomto stĺpci medzi nájdenými zarovnaniami?
  - Použite príkaz grep ako v časti C na odfiltrovanie hlavičky
  - Spojte ho s príkazom sort -g -k 3 ktorý tabuľku číselne usporiada podľa tretieho stĺpca
  - A ešte ho spojte s príkazom head -n 1 resp. tail -n 1 na vypísanie prvého alebo posledného riadku
- Výsledné príkazy a výsledky si zapíšte do protokolu

Časť E (nepovinná): Hľadanie podobných proteínov, blastp, formatdb

- Do svojho adresára DU8 si skopírujte súbory /projects/data-ppb/2010-L/drosophila/human-prot.fasta a /projects/data-ppb/2010-L/drosophila/droMelGenes-prot.fasta
  - Prvý obsahuje ľudské proteíny z databázy Swissprot a druhý proteíny z kúska genómu Drosophila melanogaster, ktorý práve študujeme.
- Skúste proteínový blast medzi týmito dvoma súbormi, pričom ľudské proteíny budú databáza, ktorú musíme najskôr sformátovať:

```
formatdb -i human-prot.fasta
blastall -p blastp -i droMelGenes-prot.fasta -d human-prot.fasta -m 9 -e 0.01 > blastp.out
```

- Zo súboru blastp.out pomocou perlu vypíšte do súboru blastp.pairs dvojice proteínov, ktoré sa na seba podobajú.
  - Vynechanie komentárov, vypísanie iba prvých dvoch stĺpcov: `grep -v '#' blastp.out | perl -lane 'print $F[0], " ", $F[1]'`
  - Pridajte správne presmerovanie výstupu
- Niektoré dvojice môžu byť v súbore viac než raz, vyfiltrujte ich pomocou sort a uniq, výsledok presmerujte do súboru blastp.unique
- S koľkými proteínmi je podobný droMel-CG8565-RA? (príkazy grep a wc)
- Dal by tento postup správne výsledky, aj keby by v súbore bol aj gén s menom napr. droMel-CG8565-RA2?

Prednáška 9

Domáca úloha 9 (5%) • Odovzdávanie D.Ú.9 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Opakovanie

- Minule: príkazy na jednoduchú analýzu textových súborov wc, grep, sort, uniq, head, tail
- Jednoriadkové programy v Perle
- Spájanie programov, presmerovanie vstupu a výstupu (|, >, <)
- Program NCBI blast na hľadanie podobností, zarovnaní medzi sekvenciami

Táto prednáška:

- Zhrnutie práce na príkazovom riadku
- Práca s procesmi
- Ďalšie bioinformatické programy
  - Budeme ich spúšťať na drosofilových sekvenciách z minulých prednášok (ako keby to bola novo-osekvenovaná sekvencia, o ktorej sa snažíme niečo zistiť).
  - Viac o tom, ako tieto programy fungujú a čo robia na predmete Metódy v bioinformatike (http://compbio.fmph.uniba.sk/vyuka/mbi/poznamky/)

Príkazy na príkazovom riadku

- Prvé je vždy meno príkazu/programu (prípadne s cestou)
- Nasleduje medzera a ďašie nastavenia/argumenty oddelené medzerami
- Ak argumenty obsahujú špeciálne znaky, napr. $, &, ;, <, > dávame ich do apostrofov
- Zvyčajne nepovinné nastavenia majú formu -písmeno, prípadne -písmeno hodnota
- Výstup programu môžeme poslať ďalšiemu programu pomocou | alebo do súboru pomocou >

Príklady z predchádzajúcich prednášok:

```
bl2seq -p blastn -i droMel.fasta -j droYak.fasta >blast.out
./priemer.pl /projects/data-ppb/du6/repeats2.txt
grep '>' /var/www/lsb09-web/drosophila/drosophilas.fasta
grep XXX subor.txt | sort | uniq | wc
```

Práca s procesmi

- Bioinformatické programy budeme spúšťať na malých hračkárskych príkladoch
- Na väčších dátach bežia dlhšie
- Aby sme zistili, čo práve beží na serveri a koľko pamäte to žerie, použijeme príkaz `top`
- Zoznam vašich bežiacich procesov pomocou `ps aux | grep vasemeno`
- Ak ste niektorý program spustili omylom, stopnete ho pomocou `kill cislo`, kde číslo procesu zistíte pomocou top alebo ps
- Ak máte dlhobežiaci program, môžete ho spustiť **na pozadí** pomocou `at now`
  - Po spustení `at now` dáte enter, potom zadáte príkaz, ktorý chcete spustiť a Ctrl-D
  - Program teraz môžete ukončiť len pomocou `kill`
  - Výstup programu dostanete po skončení e-mailom, ale lepšie je presmerovať ho do súboru
  - Dobré je presmerovať aj standard error, kam programy píšu chybové hlášky (pomocou 2> subor)
  - Napr. v at now spustíme program repeatmasker na hľadanie sekvenčných opakovaní takto:

```
meno@vyuka:~$ at now
warning: commands will be executed using /bin/sh
at> RepeatMasker droMel.fasta >vystup 2>error
at> <EOT>
job 1 at Mon Nov 29 21:16:00 2010
meno@vyuka:~$
```

Viacnásobné zarovnania, fylogenetika

- Viacnásobné zarovnania nám umožnujú zarovnať viacero sekvencií a vidieť, ktoré ich časti sú viac konzervované než iné
- Väčšinou viacnásobné zarovnania fungujú globálne, t.j. na celých vstupných sekvenciách, ktoré by preto mali byť homológy
- V cvičení budeme bežať program **muscle** [9] (http://www.drive5.com/muscle/) , môžete si vyskúšať aj ďalšie (máme nainštalované napr. clustalw, probcons a t-coffee)

- Zarovnané sekvencie môžeme použiť na zostavenie fylogenetického stromu
- Opäť existuje veľa metód, my použijeme maximum likelihood metódu z programu **phyml** [10] (http://atgc.lirmm.fr/phyml/)
- Táto metóda vychádza z pravdepodobnostného modelu evolúcie, hľadá strom, ktorý najlepšie sedí s modelom

Hľadanie opakovaní

- Sequence repeats (opakovania) tvoria veľkú časť genómov vyšších organizmov, napr. cca 50% ľudského genómu
- transposons, tandem repeats, low-complexity DNA
- Repeaty robia problémy mnohým ďalším bioinformatickým nástrojom
- Preto sa často "maskujú": nahradzujú N alebo malými písmenami
- Program RepeatMasker [11] (http://www.repeatmasker.org/) používa svoju knižnicu známych repeatov pre rôzne genómy
- Sú aj programy na hľadanie opakujúcich sa sekvencií v nových genómoch

Hľadanie génov

- My budeme používať Augustus [12] (http://augustus.gobics.de/)
- Je jeden z viacerých dostupných programov na hľadanie génov kódujúcich proteíny v DNA
- Používa štatistický model (zovšeobecnený skrytý Markovov model) na rozlíšenie kódujúcich a nekódujúcich sekvencií, signálov zostrihu a pod.
- Na nový genóm treba upraviť parametre (zložitý proces)
- Existujú parametre pre veľa genómov
- V prokaryotických genómoch bez intrónov ľahšia úloha: napr. glimmer (tigr-glimmer) [13] (http://www.cbcb.umd.edu/software/glimmer/)

- Augustus nám dá súradnice génov v **GTF formáte (http://mblab.wustl.edu/GTF22.html)**
  - Každý riadok jeden exón alebo intrón
  - Stĺpce oddelené tabulátormi, obsahujú meno skevencie, súradnice a ďalšie údaje.
  - Príklad (http://vyuka.compbio.fmph.uniba.sk/lsb09-web/drosophila/droMelGenes.gtf)
  - Používa sa aj podobný GFF formát (http://www.sanger.ac.uk/resources/software/gff/)

Profily rodín proteínov a RNA

- Existujú databázy rodín proteínov, napr. **Pfam** [14] (http://pfam.sanger.ac.uk/)
- Pre každú rodinu viacnásobné zarovnanie a pravdepodobnostný profil rodiny
- Ak nevieme nájsť homológy pre daný proteín, možno nájdeme aspoň podobný profil: senzitívnejšie vyhľadávanie
- Podľa nájdených profilov vieme usudzovať o funkcii proteínu
- My budeme používať databázu CDD [15] (http://www.ncbi.nlm.nih.gov/cdd) , ktorá združuje viacero iných databáz, napr. Pfam.
- V nej sa dá používať efektívny nástroj rpsblast

- Podobne pre rodiny RNA génov existuje databáza **Rfam** [16] (http://rfam.sanger.ac.uk/)
- Každá rodina zarovnaná nielen podľa sekvencie, ale aj sekundárnej štruktúry

Custom tracks v UCSC browseri

- UCSC genome browser (http://genome.ucsc.edu/) vie zobraziť aj vaše dáta
- Môžete ich porovnať s dátami, ktoré sú už tam, ale aj zdielať s inými
- Návod na pridávanie (http://genome.ucsc.edu/goldenPath/help/customTrack.html)

Cvičenia 9

Prednáška 9 • Odovzdávanie D.Ú.9 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Časť A: Viacnásobné zarovnania

- V svojom domovskom adresári si spravte adresár DU9 a nakopírujte si tam súbor /projects/data-ppb/2010-L/aln/myb-insects.fa
- Tento súbor obsahuje proteín MYB_DROME a jeho homológy vo viacerých drozofilách
- Spustite naňho viacnásobné zarovnanie programom muscle:

```
muscle -in myb-insects.fa  > multiple.fasta
```

- Pozrite si výsledný súbor multiple.fasta programom less, skopírujte si z neho kúsok do protokolu.
- Zarovnania si môžete prezerať programom seaview (zadajte `seaview multiple.fasta &`)

Časť B: Fylogenetické stromy

- Na základe zarovnania z časti A chceme zostrojiť strom
- Najskôr preformátujeme do formátu phylip: `readseq -f12 multiple.fasta -o=multiple.phy -a`
- Spustíme phyml: `phyml -i multiple.phy -d aa`
- Výsledný strom si pozrite programom njplot: `njplot multiple.phy_phyml_tree.txt &`
  - V literatúre sa používa tento strom: [17] (http://genome.ucsc.edu/images/phylo/dm3_15way.gif) Obsahuje síce viac druhov ako náš strom, my si však budeme všímať len vzťahy medzi druhmi, ktoré sú v oboch stromoch.
  - Vo phylogenetickom strome nezáleží, ktoré z dvoch detí nakreslíme vyššie a ktoré nižšie. V programe njplot zapnite funkciu Swap nodes a kliknutím na

znak # poprehadujte strom tak, aby sa poradie zhora dole čo najviac podobalo tomu na obrázku
    - V prípade potreby zmeňte aj zakorenenie stromu pomocou nástroja New outgroup v njplote (malo by to byť D.virilis)
    - Potom zvoľte nástroj Full tree a vo File menu Save rooted tree uložte poprehadzovaný strom do súboru
    - V polohe ktorých organizmov sa váš strom líši od obrázku vyššie?

Časť C: Hľadanie opakovaní, RepeatMasker

- V svojom adresári DU8 by ste mali mať súbor droMel.fasta, ktorý ste vytvorili v časti B DÚ8, skopírujte si ho do DU9. Ak ho nemáte, vyrobte si ho.

- Spustite RepeatMasker príkazom `RepeatMasker -species=drosophila -xsmall droMel.fasta`
- Príkaz `ls -lt` vypíše súbory v adresári od najnovších. Ktoré súbory vypísal RepeatMasker?
- Skúste príkaz `paste droMel.fasta.masked droMel.fasta | less`
    - Príkaz paste vypíše vedľa seba dva súbory, takže ich vieme ľahko vizuálne porovnať
    - V čom sa líšia (uveďte ako príklad niekoľko riadkov)?
- Pozrite si súbor droMel.fasta.out (príkazom less), uveďte ako príklad zopár riadkov. S podobným súborom sme už pracovali na DU6
- Poznámka: ďalšie nastavenia programu zistíte pomocou `RepeatMasker | less`

Časť D: Hľadanie génov, Augustus

- Spustite Augustus príkazom `augustus --species=fly droMel.fasta > augustus-genes.gtf`
- Pozrite si výsledok programom less. Koľko intrónov má prvý gén v súbpre a koľko druhý?

Časť E (nepovinná): UCSC genome browser custom track

- Výsledok z Augustusu si chceme pozrieť v UCSC genome browseri a porovnať so známymi génmi
- Z výstupného súboru augustus-genes.gtf vyextrahujme iba riadky so súradnicami exónov, start a stop kodónov (budeme to potrebovať na zobrazovanie):
    - `grep '^droMel' augustus-genes.gtf | grep -E '(CDS|codon)' > augustus-display.gtf`
- Súradnice v augustus-display.gtf sú číslované tak, že prvá báza v droMel.fasta má číslo 1. V skutočnosti je to však báza 15700001 v chromozóme X.
    - Potrebujeme súradnice poposúvať a zmeniť prvý stĺpec na chrX takto:
    - /projects/data-ppb/2010-L/programs/posun.pl <augustus-display.gtf > augustus-ucsc.gtf
- Výsledný súbor augustus-ucsc.gtf skopírujte do adresára ~/public_html/
    - (~ znamená váš domovský adresár, t.j. /home/vasemeno/)
- Nastavte pre tento súbor všetkým práva na čítanie (chmod a+r menosuboru)
    - Overte si vo firefoxe, že je prístupný na http://vyuka.compbio.fmph.uniba.sk/~vasemeno/augustus-ucsc.gtf
- Na http://genome.ucsc.edu/ zvoľte Genomes, Insect, Drosophila melanogaster, add custom tracks
    - Na ďalšej stránke vložte http://vyuka.compbio.fmph.uniba.sk/~vasemeno

/augustus-ucsc.gtf do okienka *Paste URLs or data*, stlačte *Submit*, na ďalšej stlačte *Go to genome browser*

- Do okienka position v browseri zadajte chrX:15,714,300-15,719,997 Ako sa líši Augustova predikcia tohto jedného génu s génom z databázy Flybase alebo Refseq?
- Poznámka: ak máte GTF súbor priamo na Vašom počítači, nemusíte ho dávať na webstránku, stačí ho vložiť pomocou Browse priamo na stránke add custom tracks. Teraz sme ho mali na serveri vyuka, takže by sme si ho museli najskôr pomocou scp alebo winscp stiahnuť.

Časť F (nepovinná): Profily rodín proteínov

Súbory /projects/data-ppb/aln/cdd.* obsahujú dva profily z CDD. Chceme ich výskyty vyhľadať na proteínových sekvenciách génu MYB_DROME. Nemusíte si CDD profily kopírovať, stačí ak použijete absolútnu cestu:

```
rpsblast -d /projects/data-ppb/2010-L/aln/cdd -m 9  -i myb-insects.fa -e 0.01 -o domeny.out
```

- Do protokolu si skopírujte kúsok výstupu.
- Vo výstupe je v treťom stĺpci skóre. Ppužite príkaz `sort -g -k 3 domeny.out | less` na utriedenie tohto súboru podľa tretieho stĺpca, potom vo výsledku zistite, v proteíne ktorého organizmu sa našiel výskyt s celkovo najvyššíme skóre.
- Na stránke http://www.ncbi.nlm.nih.gov/cdd do vyhľadávacieho okienka zadajte číslo nájdenej domény 29107. Akú má funkciu?
- Oprava (2010-12-06): skóre je v 12. stĺpci, t.j. príkaz má byť `sort -g -k 12 domeny.out | less`
- Poznámka: pomocou nastavenia -p F vie rpsblast hľadať aj v nukleotidových sekvenciách

Prednáška 10

Domáca úloha 10 (5%) • Odovzdávanie D.Ú.10 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=22)

Dnes sa vrátime k Perlu a ukážeme si ako z perlového programu spúšťať iné programy.

Spúšťanie programov

- Z perlového programu môžeme ľahko spustiť iné programy pomocou funkcie system:

```
#!/usr/bin/perl -w
use strict;

# spustime prikaz ls
system("ls");
```

Funkcia system tiež vráti číslo, ktoré je 0, ak program zbehol v poriadku, alebo iné číslo, ak došlo ku chybe. Toto číslo je dobré otestovať.

```
#!/usr/bin/perl -w
use strict;
```

```
# spustime prikaz ls
my $hodnota = system("ls zly_subor");
if($hodnota != 0) {
        die "Chyba pri bezani programu";
}
```

Dostávame:

```
meno@vyuka:~$ ./zle_ls.pl
ls: cannot access zly_subor: No such file or directory
Chyba pri bezani programu at ./zle_ls.pl line 7.
```

Podprogramy

- Ak by sme chceli spúšťať za sebou niekoľko programov, po spustení každého by sme mali testovať premennú $hodnota
- Aby sme stále nepísali to isté, spravíme si radšej podprogram (niečo ako makro)

```
#!/usr/bin/perl -w
use strict;

# spustime prikaz ls
spusti("ls");
spusti("ls zly_subor");

# podprogram, ktory dostane prikaz na spustenie
# vypise ho, spusti ho a detekuje chybu
sub spusti {
    my ($prikaz) = @_;
    print $prikaz, "\n";
    my $hodnota = system($prikaz);
    if($hodnota != 0) {
        die "Chyba pri bezani programu\n$prikaz";
    }
}
```

Dostávame:

```
meno@vyuka:~$ ./zle_ls.pl
ls
zle_ls.pl
ls zly_subor
ls: cannot access zly_subor: No such file or directory
Chyba pri bezani programu
ls zly_subor at ./zle_ls.pl line 15.
```

Zoznam súborov

Nasledujúci príkaz vypíš mená súborov s príponou .pl v aktuálnom adresári (t.j. zhruba to isté ako ls *.pl)

```
#! /usr/bin/perl -w

use strict;

my @subory = glob("*.pl");
foreach my $subor (@subory) {
    print $subor, "\n";
}
```

Dostávame:

```
meno@vyuka:~$ ./zoznam.pl
zle_ls.pl
zoznam.pl
```

- glob("*.pl") vráti zoznam všetkých súborov s koncovkou .pl
- uložíme si ich do poľa @subory
- cyklus foreach my $subor (@subory) { ... } spúšťa príkazy pre každý prvok poľa, pričom hodnotu tohto provku uloží do premennej $subor

Stromy s Phyml

- Nasledujúci program spustí na každý súbor s príponou .fa v aktuálnom adresári muscle na viacnásobné zarovnanie, zmení formát a spustí phyml (tieto tri kroky ste robili ručne na DÚ 9).
- V každom .fa súbore by sme teda mali mať sekvencie zodpovedajúcich si homologických proteínov z rôznych druhov.

```
#!/usr/bin/perl -w
use strict;

my @subory = glob("*.fa");
foreach my $fasta (@subory) {
    #mena suborov
    my $zarovnanie = $fasta . ".aln";
    my $zarovnanie2 = $fasta . ".phy";

    #spustanie programov
    spusti("muscle -in $fasta  > $zarovnanie");
    spusti("readseq -f12 $zarovnanie -o=$zarovnanie2 -a");
    spusti("phyml -i $zarovnanie2 -d aa");
}

# podprogram, ktory dostane prikaz na spustenie
# vypise ho, spusti ho a detekuje chybu
sub spusti {
    my ($prikaz) = @_;
    print $prikaz, "\n";
    my $hodnota = system($prikaz);
    if($hodnota != 0) {
        die "Chyba pri bezani programu\n$prikaz";
    }
}
```

- Všimnite si, že medzi úvodzovky môžeme písať aj premenné, Perl dosadí hodnotu.

Stromy s programom Phylip

- Balík Phylip obsahuje veľa programom na fylogentické stromy, čaká však, že ho bude užívateľ riadiť z klávesnice
- Namiesto toho môžeme klávesy zapísať do súboru `prikaz` a presmerovať Phylipu pomocou <
- Vstup berie zo súboru infile alebo intree, výstup píše do outfile a/alebo outtree (tie nesmú existovať)

```
#! /usr/bin/perl -w

use strict;
```

```
my @subory = glob("*.fa");
foreach my $fasta (@subory) {
    my $zarovnanie = $fasta . ".aln";
    my $zarovnanie2 = $fasta . ".phy";
    my $matica = $fasta . ".dist";
    my $strom = $fasta . ".tree";

    spusti("muscle -in $fasta  > $zarovnanie");
    spusti("readseq -f12 $zarovnanie -o=$zarovnanie2 -a");
    spusti("echo Y > prikaz");
    unlink("outfile");
    spusti("cp $zarovnanie2 infile");
    spusti("phylip protdist < prikaz");
    spusti("mv outfile $matica");

    unlink("outfile");
    unlink("outtree");
    spusti("cp $matica infile");
    spusti("phylip neighbor < prikaz");
    spusti("mv outtree $strom");
}

spusti("cat *.tree > stromy");
spusti("cp stromy intree");
unlink("outtree");
unlink("outfile");
spusti("echo \"D\n2\nP\nF\nY\n\" > prikaz");
spusti("phylip treedist < prikaz");
spusti("mv outfile stromy.rozdiely");

sub spusti {
    my ($prikaz) = @_;
    print $prikaz, "\n";
    my $hodnota = system($prikaz);
    if($hodnota != 0) {
        die "Chyba pri bezani programu\n$prikaz";
    }
}
```

- Perlová funkcia unlink maže súbory.
- Linuxový príkaz echo vypíše zadaný výraz a príkaz cat vypíše obsah zadaných súborov.
- V príkaze `spusti("echo \"D\n2\nP\nF\nY\n\" > prikaz");` používame `\"` ako úvodzovky vo vnútri úvodzoviek.
- Spustíme najprv zarovnanie pomocou muscle a konvertovanie formátu pomocou readseq
- Potom spočítame vzdialenosti medzi sekvenciami pomocou Phylipa a ďalším zbehnutím Phylipa spočítame z týchto vzdialenosí strom metódou spájania susedov (Neighbor joining)
- Toto opakujeme pre každý .fa súbor.
- Nakoniec spustíme Phylip ešte raz, aby sme dostali tabuľku porovnávajúcu jednotlivé stromy.

Námety na projekt

Tu je niekoľko námetov na projekty týkajúce sa Perlu

- Naučiť sa funkcie na prácu s reťazcami a regulárnymi výrazmi v Perle. Viac sa dozviete napr. pomocou man perlretut (http://perldoc.perl.org/perlretut.html) , man perlre (http://perldoc.perl.org/perlre.html) , [18] (http://perldoc.perl.org/index-functions-by-cat.html) . Príklady použitia:
  - Napr. v programe vyššie robíme mená súborov volaco.fa.tree a pod., ale možno by bolo lepšie len volaco.tree.
  - Z DNA sekvencie spraviť sekvenciu pre komplementárne vlákno.

    - Nahradiť v DNA sekvencii T na U, čím ju premeníte na RNA.
    - Kontrola správnosti formátu súborov, rôzne zmeny vo formáte, ...
- Naučiť sa pracovať s knižnicou Bioperl (http://www.bioperl.org/)
- Naučiť sa pracovať v Perle s nastaveniami programu. Viac sa dozviete napr. pomocou man Getopt::Std (http://perldoc.perl.org/Getopt/Std.html)
    - Môžete spraviť program, ktorý kombinuje niekoľko programov z cvičení a podľa nastavení zadaných užívateľom sa rozhodne, ktorý presne spustí a s akými nastaveniami.

Cvičenia 10

Prednáška 10 • Odovzdávanie D.Ú.10 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=22)

Časť A

- Skopírujte si adresár /projects/data-ppb/DU10/ do svojho domovského adresára (napr. v domovskom adresári zadáte `cp -r /projects/data-ppb/DU10/ .`).
- Presuňte sa do DU10 a spustite príkaz ls (výsledok do protokolu)
- Každý z *.fa súborov obsahuje sekvencie jedného proteínu z rôznych kvasiniek. Pozrite si jeden z nich pomocou príkazu less, zopár riakov skopírujte do protokolu.
- Spustite program ./stromy_phylip.pl (nekopírujte výstup do protokolu, je ho veľa)
- Aké súbory začínajúce na ATP6 program vytvoril? (príkaz a výsledok do protokolu) Čo je v každom z nich? (nemusíte uvádzať úryvky, popíšte slovne, čo si myslíte, že je význam každého súboru)

Časť B

- V časti A by vám mal vzniknúť aj súbor stromy.rozdiely, ktorý obsahuje tabuľku rozdielov medzi jednotlivými stromami, ktoré sme vyrobili programom stromy_phylip.pl. Namiesto mien proteínov sú v tabuľke len čísla, sú však očíslované podľa abecedy, t.j. ATP6 je prvý, potom COB, COX1 atď. Číslo 0 vo vnútri tabuľky znamená, že príslušné stromy sa topológiou nelíšia (môžu sa líšiť dĺžkami hrán). Ktorý strom sa podľa tejto tabuľky líši od ostatných?
- Ak chcete, môžete sa pozrieť, ako sa tieto stromy líšia programom njplot (nebodované).

Časť C

- Do programu ./stromy_phylip.pl pridajte aspoň na päť miest komentáre vysvetľujúce, čo daná časť programu robí. Komentár začína znakom #.
    - Zmenený program skopírujte do protokolu.

Časť D (nepovinná)

Do programu ./stromy_phylip.pl pridajte príkazy, ktoré každý strom uložia ako obrázok do .bpm súboru. Ako na to:

- Použite linuxový príkaz `phylip drawtree < prikaz` (ktorý spustíte v Perle pomocou podprogramu spusti)

- Predtým ako sa spustí Phylip, treba mu nakopírovať strom do súboru intree, zmazať súbor plotfile a do súboru prikaz uložiť "P\nW\n1500\n1500\nV\nN\nY"
- Po skončení Phylipa treba výsledok skopírovať zo súboru plotfile do súboru s menom typu volaco.fa.bmp, kde volaco je meno proteínu.
- Výsledné obrázky si môžete pozrieť príkazom `eog *.bmp` (listujete medzi nimi napr. medzerou alebo šípkami)
- Do protokolu zapíšte zmenenú časť programu (nemusíte celý).

Prednáška 11

Domáca úloha 11 (3%) • Odovzdávanie D.Ú.11 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=23) • Odovzdávanie D.Ú.12 plán projektu (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Dnes:

- Asociatívne polia
- Zhrnutie
- Pre záujemcov: otváranie súborov, práca s nastaveniami, ukážka z prednášky 1

Asociatívne polia

- Polia, ktoré sme videli doteraz majú políčka očíslované 0,1,2,...
- Asociatívne pole, zvané aj hash, má políčka nazvané ľubovoľnými reťazcami
- Príklad: použijeme výstup z RepeatMaskera a spočítame pre každý typ opakovania (napr. Low_complexity, Simple_repeat a pod), koľko takých opakovaní je v súbore

```
#!/usr/bin/perl -w
use strict;

#asociatívne pole, ktoré má políčka pomenované podľa typu
my %pocet;

while(my $riadok=<>) {
    #načítaný riadok rozbijeme na stĺpce
    my @stlpce = split "\t", $riadok;
    my $typ = $stlpce[11];
    #zvýšime hodnotu uloženú v $pocet{$typ} o 1
    $pocet{$typ}++;
}

#dočítali sme súbor, vypíšeme, čo sme našli
foreach my $typ (keys %pocet) {
    print $typ, " ", $pocet{$typ}, "\n";
}
```

Porovnanie polí a asociatívnych polí

```
#Vytvorenie pola a
my @a;
#Vytvorenie asociativneho pola
my %a;

#Pristup k prvku 0 v poli @a
$a[0]
#Pristup k prvku "X" v asociativnom poli %a
$a{"X"}
```

```
#Vypisanie vsetkych prvkov pola @a
for(my $i=0; $i<scalar(@a); $i++) {
  print $i, " ", $a[$i], "\n";
}

#Vypisanie vsetkych prvkov asociativneho pola %a
foreach my $meno (keys %a) {
  print $meno, " ", $a{$meno}, "\n";
}
```

Nastavenia programu

- Všetky nastavenia, mená súborov a pod, ktoré užívateľ zadal na príkazovom riadku máme v poli @ARGV
- V premennej $0 máme meno skriptu, ktorý spúšťame
- Tu je úryvok z programu, ktorý skontroluje, či užívateľ zadal dve mená súborov a ak áno, uloží ich do premenných $meno1 a $meno2:

```
if(scalar(@ARGV)!=2) {
    die "Zadaj mena dvoch suborov!\n $0 subor1 subor2\n";
}
my $meno1 = $ARGV[0];
my $meno2 = $ARGV[1];
print "Zadali ste $meno1 a $meno2\n";
```

A toto vypíše:

```
meno@vyuka:~./skontroluj.pl
Zadaj mena dvoch suborov!
 ./skontroluj.pl subor1 subor2

meno@vyuka:~$ ./skontroluj.pl text.txt iny.txt
Zadali ste text.txt a iny.txt
```

Otváranie súborov

- Doteraz sme vedeli čítať iba súbor zadaný na príkazovom riadku za menom skriptu pomocou <>
- Vypisovať sme vedeli iba na obrazovku, čo sme mohli presmerovať do súbpru pomocou > na príkazovom riadku
- Perlový skript však môže čítať alebo prepisovať viacero súborov.
- Súbor na čítanie otvoríme takto:

```
my $subor;
open $subor, "<", $meno;
while(my $riadok = <$meno>) {
  #prikazy na spracovanie riadku
}
close($subor);
```

- Premenná $subor obsahuje otvorený súbor ("záložku", kde práve sme).
- Príkaz open súbor otvorí, príkaz close zavrie.
- Znak "<" určuje, že súbor budeme čítať.
- V cykle while medzi <> dáme premennú $subor

- Je dobré skontrolovať, či sa súbor podarilo otvoriť (či existuje, máme k nemu práva atď)

```
my $subor;
open $subor, "<", $meno or die "Neviem otvorit $meno";
```

- Podobne otvoríme súbor aj na zapisovanie, iba použijeme znak ">"
- Pozor, pôvodný obsah súboru sa premaže!

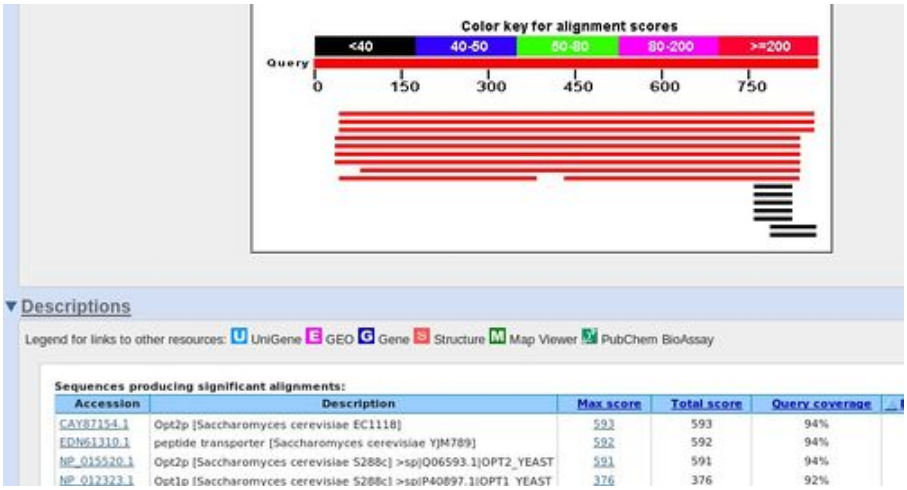
```
my $subor;
open $subor, ">", $meno or die "Neviem otvorit $meno";
print $subor "Hello world!\n";
close($subor);
```

Ukážka z prvej prednášky

- Na prvej prednáške sme si ukázali príklad, ako existujúcimi nástrojmi a krátkym programom v Perle vyriešiť jednoduchú bioinformatickú úlohu.

Úryvok z prednášky:

- Predstavme si, že máme 50 proteínov o ktorých chceme zistiť niečo viac, napr, čo robí ich najbližší homológ z kvasinky Saccharomyces cerevisiae
- Môžeme robiť ručne jeden po druhom na http://blast.ncbi.nlm.nih.gov/



- Alebo si stiahneme všetky proteíny z Saccharomyces cerevisiae, napr. tu [19] (http://downloads.yeastgenome.org/sequence/genomic_sequence/orf_protein/)
- Potom spustíme blast

```
formatdb -i yeast-prot.fa
blastall -p blastp -i prots.fa -d yeast-prot.fa -m 9 -e 0.01 > blastp.out
```

- Výsledky z blastu spracujeme krátkym programom v Perle do tabuľky, ktorú si môžeme otvoriť napr. v Exceli

```
./spracuj.pl yeast-prot.fa blastp.out > vysledok.csv
```

- Súbory sú v adresári `/projects/data-ppb/ukazka/`
- Ak náš zoznam 50 proteínov niekedy zmeníme, za pár sekúnd vieme vytvoriť novú tabuľku.

Súbor yeast-prot.fa obsahuje všetky S. cerevisiae proteíny. Je vo fasta formáte, na riadku začínajucom > je meno proteínu a jeho popis, potom nasleduje sekvencia. Tu je úryvok:

```
>YAL013W DEP1 SGDID:S000000011, Chr I from 129271-130533, Verified ORF, "Transcriptional modulator involved in r
MSQQTPQESEQTTAKEQDLDQESVLSNIDFNTDLNHNLNLSEYCISSDAGTEKMDSDEEK
SLANLPELKYAPKLSSLVKQETLTESLKRPHEDEKEAIDEAKKMKVPGENEDESKEEEKS
QELEEAIDSKEKSTDARDEQGDEGDNEEENNEEDNENENEHTAPPALVMPSPIEMEEQRM
TALKEITDIEYKFAQLRQKLYDNQLVRLQTELQMCLEGSHPELQVYYSKIAAIRDYKLHR
AYQRQKYELSCINTETIATRTFIHQDFHKKVTDLRARLLNRTTQTWYDINKERRDMDIVI
PDVNYHVPIKLDNKTLSCITGYASAAQLCYPGEPVAEDLACESIEYRYRANPVDKLEVIV
DRMRLNNEISDLEGLRKYFHSFPGAPELNPLRDSEINDDFHQWAQCDRHTGPHTTSFCYS
*
>YAL014C SYN8 SGDID:S000000012, Chr I from 129020-128253, reverse complement, Verified ORF, "Endosomal SNARE rel
MDVLKLGYELDQLSDLVEERTRLVSVLKLAPTSNDNVTLKRQLGSILELLQKCAPNDELI
SRYNTILDKIPDTAVDKELYRFQQQVARNTDEVSKESLKKVRFKNDDELTVMYKDDDEQD
EESPLPSTHTPYKDEPLQSQLQSQSQPQPPQPMVSNQELFINQQQQLLEQDSHLGALSQS
IGRTHDISLDLNNEIVSQNDSLLVDLENLIDNNGRNLNRASRSMHGFNNSRFKDNGNCVI
ILVLIVVLLLLLLVL*
```

Súbor prots.fa obsahuje naše proteíny, ktoré chceme skúmať, v podobnom formáte:

```
>gnl|GLV|YALI0A00198g
MLNNEEDTQQQLTHTLLPALMANWFSMILRHGVMLSGLTVTIYRLQNQNLSPGEQGRRVM
ELVACLAAGILPLWTPPKISAAQLHLFQVSPYIFTAAILLASTLLLGVCQCWLLSVHPAA
FVLVLLLNSFLVNEIKWSEAFFASNWLPESQYYETIEKSKQLTILSTLISSGLVALFKTA
WFYLVLFAAVLYVSYMMAFVVDARLENQTKYKQMLEESEDFAPLQEWNDDHSSGVYAVIR
ERNWAAWGVYYFFQYL
>gnl|GLV|YALI0A00212g
MKLPTEIVAQICASLDLESLLQLSYTNARLRAVASNLFQQKIQASWPWITDCSDWYQEGI
KIVLARRRMREKSPMSEELPVVQVARHLIHVNEPVPEEAILLGPETKYMDDDHDLYFRTA
SLTRINPRNMANPLSYHDPLCMDVDDFLQLTGTKNAHGKTIVTHTFMGRYWIFAYDEGGG
VLEAPKLAAIDNWIYCYFQTREDGYTVIKHHCITDGVKDVPDYLVSLAKSEYGSTYDLDF
FFSRENVFVIDKLVFGNEDDFKPVYFRLLNMETGATVPLCFVDKSDTHVVEYEAFVDYMR
YLWLDGPFFYIYSGADEMIIVVDVKEGLLTISLQDNLGGE
```

Súbor blastp.out obsahuje výsledky, ktoré vypísal blast, pričom každý ďalší proteín z prots.fa začína hlavičkou niekoľkých riadkov začínajúcich #, potom ide tabuľka nájdených podobností. Podobný súbor sme už spracovávali na DÚ8.

```
# BLASTP 2.2.21 [Jun-14-2009]
# Query: gnl|GLV|YALI0A00110g
# Database: yeast-prot.fa
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. sta
gnl|GLV|YALI0A00110g    YPR194C 39.41   812     442     16      75      866     89      870     2e-153   537
gnl|GLV|YALI0A00110g    YJL212C 31.96   776     471     18      89      843     33      772     8e-97    350
# BLASTP 2.2.21 [Jun-14-2009]
# Query: gnl|GLV|YALI0A00132g
# Database: yeast-prot.fa
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. sta
gnl|GLV|YALI0A00132g    YNL209W 81.73   613     110     1       1       611     1       613     0.0      998
gnl|GLV|YALI0A00132g    YDL229W 81.57   613     111     1       1       611     1       613     0.0      996
gnl|GLV|YALI0A00132g    YER103W 62.15   605     223     5       9       609     4       606     0.0      717
# BLASTP 2.2.21 [Jun-14-2009]
# Query: gnl|GLV|YALI0A00176g
# Database: yeast-prot.fa
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. sta
# BLASTP 2.2.21 [Jun-14-2009]
# Query: gnl|GLV|YALI0A00198g
# Database: yeast-prot.fa
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. sta
```

Pre každý Query proteín chceme by sme chceli len prvý riadok z jeho tabuľky výsledkov (najnižšia E-value) a navyše by sme chceli aj dlhší popis, ktorý v tabuľke nie je, iba v pôvodnom fasta súbore. T.j. vypíšeme nejaký takýto výstup:

```
gnl|GLV|YALI0A00110g    YPR194C 2e-153  OPT2 SGDID:S000006398, Chr XVI from 926933-924300, reverse complement, V
```

```
gnl|GLV|YALI0A00132g    YNL209W 0.0     SSB2 SGDID:S000005153, Chr XIV from 252060-253901, Verified ORF, "Cytopl
gnl|GLV|YALI0A00154g    YKL077W 2e-07   YKL077W SGDID:S000001560, Chr XI from 291097-292275, Uncharacterized ORF
gnl|GLV|YALI0A00176g
gnl|GLV|YALI0A00198g
gnl|GLV|YALI0A00212g
```

Toto robí perlový program spracuj.pl.

```
#! /usr/bin/perl -w

use strict;

#pole @ARGV by malo obsahovat mena dvoch suborov:
# fasta subor pre nasu databazu proteinov z S. cerevisiae
# a vystup z blast-u ktory porovnaval nezname proteiny s db
if(scalar(@ARGV)!=2) {
    die "Zadaj mena dvoch suborov!\n $0 fasta blast > vysledok\n";
}
my $meno_fasta = $ARGV[0];
my $meno_blast = $ARGV[1];

# otvorime subor s fastou, ak sa nepodari, vypiseme chybu
my $subor;
open $subor, "<", $meno_fasta or die "Neviem otvorit $meno_fasta";

# zo suboru nacitame pre kazdy protein meno a popis
my %slovnik;
while(my $riadok = <$subor>) {
    #testujeme ci riadok zacina na > (pouzitie regularnych vyrazov)
    if($riadok =~ />(\S+)\s+(.*)\s*$/) {
        my $meno = $1;
        my $popis = $2;
        $slovnik{$meno} = $popis;
    }
}
close $subor;

# otvorime subor s vystupom z blastu
open $subor, "<", $meno_blast or die;
#pamatame si meno aktualneho proteinu (query)
# (ak sme k nemu este nic nevypisali)
my $meno = "";
while(my $riadok = <$subor>) {
    #ak riadok obsahuje meno novej query, ulozime si ho do $meno
    # (test pomocou regularnych vyrazov)
    if($riadok =~ /^# Query: (\S+)/) {
        #ak sme k predchadzajucemu menu nic nenasli,
        # vypiseme prazdne stlpce
        if($meno ne "") {
            print "$meno\t\t\t\n";
        }
        $meno = $1;
    }
    # ak mame riadok tabulky o podobnosti
    # a este sme k $meno nic nevypisali, vypiseme udaje z riadku
    elsif(!($riadok =~ /^#/) && $meno ne "") {
        #rozlozime riadok na stlpce, kontrola
        my @stlpce = split "\t", $riadok;
        if(@stlpce != 12 || $stlpce[0] ne $meno) {
            die "Chyba v riadku $riadok";
        }
        #meno proteinu z db by malo byt v slovniku
        my $meno2 = $stlpce[1];
        if(! exists $slovnik{$meno2}) {
            die "Chyba popis k proteinu $meno2";
        }
        my $popis = $slovnik{$meno2};
        #vypisme meno, meno proteinu z db, jeho popis, E-value
        print $meno, "\t", $meno2, "\t", $stlpce[10], "\t", $popis, "\n";
        # k proteinu $meno sme uz nieco vypisali, vymazme ho
        $meno = "";
    }
}
close $subor;
```

```
# na konci nam mohol zostat posladny nevypisany protein
if($meno ne "") {
    print "$meno\t\t\t\n";
}
```

- Na príkazovom riadku dostane mená dvoch programov (proteínová db vo fasta formáte a výstup z blastu)
- Najprv načíta prvý riadok a vytvorí si slovník (pre každé meno proteínu v S. cerevisiae jeho popis v asociatívnom poli)
- Potom načíta výstup z blastu
  - Vždy keď vidí riadok typu `# Query: gnl|GLV|YALI0A00198g` zapamätá si meno do premennej $meno
  - Keď potom uvidí riadok tabuľky (nezačínujúci #), vypíše údaje o príslušnom S.cerevisae proteíne za pomoci slovníka
  - Keď už vypísal riadok pre konkrétne meno, $meno vymaže
  - Keď príde na začiatok ďalšej query a meno nie je vymazané, tento proteín nemá homológ, tak vypíšeme prázdne stĺpce do výstupu.

Zhrnutie semestra

- Základy používania systému Linux (grafického režimu aj príkazového riadku)
- Linuxové príkazy na prácu so súbormi, práva súborov (ls, cp, rm, chmod, ...)
- Linuxové príkazy na spracovanie textových súborov (grep, wc. sort, uniq, ...)
- Používanie bioinformatických nástrojov (blast, RepeatMasker, Augustus, muscle, phyml, ...)
- UCSC genome browser
- Základy programovania v jazyku Perl (mierne meniť existujúce programy, písať jednoduché krátke programy)

Nadväzujúce predmety na FMFI:

- 2-AIN-501 Metódy v bioinformatike (http://compbio.fmph.uniba.sk/vyuka/mbi/) (viac o tom aké sú princípy fungovania bioinformatických nástrojov ako napr. blast) ZS, 4 hodiny, 6 kreditov
- 2-AIN-503, 2-AIN-504, 2-AIN-251, 2-AIN-252 Seminár z bioinformatiky (http://compbio.fmph.uniba.sk/vyuka/seminar/) (čítame nové články z bioinformatickej literatúry) ZS/LS, 2 hodiny, 2 kredity
- 1-MAT-130 Programovanie (1) (http://edi.fmph.uniba.sk/%7Esalanci/C/index.html) (programovanie v C++ pre matematikov) ZS, 4 hodiny, 4 kredity

Koniec semestra:

- Termín domácich úloh 10 a 11, projektu je 17.1.2011 o 23:55.
- Termín návrhu projektu (DÚ12) je 14.12.2010 do 23:55.
- Do AIS2 vypíšem termíny na ústnu skúšku ohľadom projektu a zapisovanie známky (po 17.1.), zapíšte sa.

Upozornenie:

- Po skončení skúškového vám ešte nejaký čas bude fungovať konto na serveri aj v Moodli, potom však bude zrušené
- Vaše DÚ z Moodlu vám pošlem e-mailom, súbory zo servera si v prípade záujmu stiahnite pomocou WinSCP

Cvičenia 11

Prednáška 11 • Odovzdávanie D.Ú.11 (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=23) • Odovzdávanie D.Ú.12 plán projektu (http://vyuka.compbio.fmph.uniba.sk/ppbmoodle/mod/assignment/view.php?id=21)

Časť A

- Vytvorte si adresár DU11 a domácu úlohu robte v ňom.
- Prvý program z prednášky 11 vypíše pre každý typ opakovania počet riadkov tohto typu v súbore. Uložte si ho do súboru pocty.pl a spustite ho na súbore /projects/data-ppb/du6/repeats2.txt v ktorom sú všetky opakovania v chromozóme 2L Drosophily melanogaster (použite príkaz ./pocty.pl /projects/data-ppb/du6/repeats2.txt). Výsledok si skopírujte do protokolu.

Časť B

- Zmeňte program tak, aby namiesto počtu pre každý typ opakovania vypísal priemernú dĺžku opakovaní tohto typu. Ako na to:
  - Priemer dĺžok všetkých opakovaní dokopy sme počítali na prednáške 6.
  - Teraz namiesto premenných $pocet a $sucet potrebujeme asociatívne polia %pocet a %sucet, ktoré majú jednu hodnotu pre každý typ opakovania.
  - Asociatívne pole %pocet už v programe je, stačí dorobiť %sucet. Vždy ho chceme zvýšiť o dĺžku opakovania, pomocou príkazu `$sucet{$typ}+=$dlzka;` pričom premennú $dlzka spočítame rovnako ako na prednáške 6.
  - Samozrejme treba zmeniť aj časť programu, kde vypisujeme výsledky.
- Zmenený program a jeho výsledky si zapíšte do protokolu. Ktorý typ opakovaní je v priemere najdlhší?

Časť C (nepovinná)

- Rozšírte program pocty.pl tak, aby v prehľadnej tabuľke vypisoval pre každý typ opakovania nasledujúce údaje:
  - počet opakovaní tohto typu
  - ich priemernú dĺžku
  - koľko percent chromozómu 2L pokrývajú (celý chromozóm 2L má dĺžku 23011544)
- Namiesto príkazu print použite príkaz printf, ktorý dovoľuje zarovnávať stĺpce a zaokrúhľovať desatinné čísla. Nasledujúci príkaz vypíše obsah premenných $typ, $pocet{$typ}, $priemer, $pokrytie v rozumnom formátovaní:

```
printf "%-15s %8d %10.1f %5.1f%%\n", $typ, $pocet{$typ}, $priemer, $pokrytie;
```

- Použitie príkazu printf nie je úplne jednoduché. Prvý argument je formátovací reťazec, ktorý určuje šírku stĺpcov, počet desatinných miest a pod. Ďalšie argumenty ($typ, $pocet{$typ},...) sú vypisované hodnoty. V prípade záujmu sa viac dozviete pomocou príkazov `perldoc -f printf` resp. `perldoc -f sprintf`).