38. Složený příkaz case a poziční parametry (cap)

Úkol: Napište skript:

```
/home/       /pv004lab/ukol_cap_     .sh
```

který se bude spouštět s jedním až třemi argumenty na příkazovém řádku. První argument musí být vždy zadán a určí provedenou operaci. Možné hodnoty prvního argumentu jsou:

- `verze` nebo `-v`
  Vypíše na standardní výstup:
  ```
  verze 363
  ```
  a úspěšně skončí.
- `celkem` nebo `-a`
  V souboru zadaném druhým argumentem spočítá počet řádků a předá číslo na standardní výstup a úspěšně skončí (pokud soubor existuje). Pokud soubor zadaný druhým řádkem neexistuje, musí skript skončit neúspěšně (tj. s nenulovým návratovým kódem, použijte např. `exit 1`) a současně musí předat chybové hlášení (libovolné, neprázdné) na standardní chybový výstup. Počet řádků získejte příkazem:
  ```
  wc -l < "cesta_k_souboru"
  ```
- `vybrane` nebo `-s`
  V souboru zadaném druhým argumentem spočítá počet řádků obsahující řetězec zadaný třetím argumentem, předá číslo na standardní výstup a úspěšně skončí. Zkontrolujte druhý argument stejně jako v předchozím případě. Pokud není zadán neprázdný třetí argument, ukončete skript neúspěšně a předejte chybové hlášení na standardní chybový výstup. Počet řádků získejte příkazem:
  ```
  grep "vzorek" "cesta_k_souboru"|wc -l
  ```
- `nevybrane` nebo `-r`
  Stejně jako předchozí, jen se počítají řádky neobsahující vzorek:
  ```
  grep -v "vzorek" "cesta_k_souboru"|wc -l
  ```

Není-li zadán první argument nebo je zadán první argument jiný, musí skript skončit neúspěšně a současně musí předat chybové hlášení na standardní chybový výstup. Musíte použít příkaz **case**.

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Složený příkaz case a poziční parametry (cap)

```
#!/bin/bash

case $1 in
verze|-v) echo "verze 363"; exit 0;;
celkem|-a) if test -e $2;
        then wc -l < "$2"; exit 0;
        else echo "file doesnt exist" >&2; exit 1; fi;;
vybrane|-s) if test -e $2;
        then if test -n "$3";
                then grep "$3" "$2"|wc -l; exit 0; fi;
        else echo "file doesnt exist" >&2; exit 1; fi;;
nevybrane|-r) if test -e $2;
        then if test -n "$3";
                then grep -v "$3" "$2"|wc -l; exit 0; fi;
        else echo "file doesnt exist" >&2; exit 1; fi;;
esac
echo "first parameter doesnt exist" >&2; exit 1
```

39. Numerické porovnání (nup)

Úkol: Napište skript, který budete volat:

```
/home/      /pv004lab/ukol_nup_      .sh cesta_k_souboru 999
```

První argument musí být cesta k existujícímu souboru. Druhý argument musí být celé číslo.
Pokud je zadán chybný počet argumentů, vypište na standardní chybový výstup chybu 7 6 a skript ukončete s návratovým kódem 1.

Pokud soubor neexistuje nebo pokud druhý argument není korektní číslo, ukončete s návratovým kódem 1 **bez** hlášení chyby na standardní chybový výstup.

Pro zjištění, zda je druhý argument korektní číslo, použijte např. konstrukci `[ $2 -eq $2 ]`, která vrátí 1 (chyba), pokud neporovnává korektní čísla. Musíte potlačit výstup na stderr přesměrováním do `/dev/null`.

Skript má na standardní výstup předat velikost souboru v bajtech v případě, že velikost souboru je větší nebo rovná druhému argumentu. Pokud je velikost menší, skript na standardní výstup předá číslo 0.

Pro zjištění velikosti souboru v bajtech použijte např. příkaz

```
stat -c %s cesta_k_souboru
```

Pro detaily příkazu **stat** vizte `man stat`.
Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Numerické porovnání (nup)

```
#!/bin/bash
if test "$#" -ne 2;
        then echo 7    6 >&2; exit 1; fi
if test -e $1 && test $2 -eq $2 2>/dev/null
        then if test $(stat -c %s $1) -ge $2;
                then stat -c %s $1;
                else echo 0; fi;
        else exit 1; fi;
```

40. Expanze proměnných a odstranění sufixu (fo2)

Úkol: Napište skript:

```
/home/      /pv004lab/ukol_fo2_     .sh
```

který přejmenuje všechny soubory nebo adresáře zadané argumenty při spouštění skriptu ze jména končícího příponou ".4 " na jméno končící příponou ".a ". Pokud argumentem zadaný soubor nebo adresář nekončí ".4 ", pak příponu ".a " za jméno přidejte.
Musíte použít příkaz **for**.

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Expanze proměnných a odstranění sufixu (fo2)

```
#!/bin/bash
for i;
do if test -e $i && test $i = 4{i%.4  };
        then mv $i ${i}.a  ;
        else mv $i ${i%.4  }.a  ; fi; done
```

41. Počet pozičních parametrů skriptu (poa)

Úkol: Napište skript:

```
/home/      /pv004lab/ukol_poa_     .sh
```

který skončí s chybovým návratovým kódem a na standardní chybový výstup předá libovolné neprázdné chybové hlášení v případě, že skript **ne**byl spuštěn se zadanými 4 pozičními parametry (argumenty). V opačném případě skončí s nulovým návratovým kódem a na standardní výstup předá řetězec:

```
ok b                5
```

Požádejte kliknutím o ověření splnění úkolu.

```
#/bin/bash
if test "$#" -eq 4;
        then echo "ok b                    5"; exit 0;
        else exho "isjkdu" >&2; exit 1; fi
```

42. Použití jména skriptu a symbolický odkaz (pjs)

Úkol: Napište skript:

```
/home/      /pv004lab/vytvorit
```

Vytvořte symbolický odkaz:

```
/home/x     /pv004lab/smazat
```

který bude ukazovat na existující skript `/home/      /pv004lab/vytvorit`.
Ve skriptu `vytvorit` vypisujte na standardní výstup obsah proměnné `$0`.

Dále ve skriptu do proměnné `JMENO` uložte pouze jméno skriptu (bez cesty) z proměnné `$0`. Pro získání jména skriptu z cesty použijte příkaz

```
basename "$0"
```

pomocí příkazové substituce standardní výstup příkazu `basename` vložte do proměnné `JMENO`.
Ve skriptu `vytvorit` vypisujte na standardní výstup obsah proměnné `JMENO`.

Ověřte si, že i když skript spouštíte různými způsoby:

```
/home/      /pv004lab/vytvorit
/home/      /pv004lab/smazat
./vytvorit
./smazat
```

máte stále v proměnné `JMENO` korektní jméno bez cesty.
Následně se ve skriptu rozhodněte podle obsahu proměnné `JMENO`, jakou funkci máte provádět.

- Pokud byl skript spuštěn jménem `vytvorit`,
  vypište na standardní výstup "`Skript 5    9 bude vytvářet.`".
- Pokud byl skript spuštěn jménem `smazat`,
  vypište na standardní výstup "`Skript 5    9 bude mazat.`".

Požádejte kliknutím o ověření splnění úkolu.

```
#!/bin/bash
JMENO=$(basename "$0")
echo $JMENO
if test $JMENO = "vytvorit";
        then echo "Skript 5    9 bude vytvářet."; fi
if test $JMENO = "smazat";
        then echo "Skript 5    9 bude mazat."; fi
```

45. Aritmetická expanze (aex)

Úkol: Napište skript:

```
/home/      /pv004lab/ukol_aex_     .sh
```

který bude spouštěn s libovolným počtem argumentů. Každý argument bude celé číslo. Argument vynásobte jeho pořadím (první argument má pořadí 1) a součiny sečtěte. Skript na standardní výstup nechť předá řádek obsahující řetězec `2    4c`, jednu mezeru a za ní součet součinů, tj. výsledek.
Skript spuštěný bez argumentů nechť předá na standardní výstup

```
2  4c 0
```

Požádejte kliknutím o ověření splnění úkolu.

```
#!/bin/bash
if test $# -eq 0;
        then echo "2[redacted]4c 0"; exit 0; fi
COUNT=0
PLACE=1
for i;
        do COUNT=$(($COUNT + $i*$PLACE)); PLACE=$(($PLACE + 1)); done
echo "2[redacted]4c $COUNT"
```

46. Zpracování pozičních parametrů a aritmetická expanze (sk2)

Úkol: Napište skript:

```
/home/      /pv004lab/ukol_sk2_     .sh
```

který vytvoří soubor zadaný prvním argumentem, a naplní jej obsahem souborů zadaných druhým a každým dalším argumentem, přičemž před každý zkopírovaný obsah vloží do cílového souboru řádek obsahující pořadové číslo kopírovaného souboru, jméno kopírovaného souboru (bez adresářů, lomítek) a konkrétní řetězec ve tvaru:

```
-- 1. soubor.txt c                f
```

Postup řešení: První argument (poziční parametr) si zkopírujte do vlastní proměnné. Potom argumenty (poziční parametry) posuňte o jeden doleva příkazem `shift`. Všechny další argumenty zpracujte složeným příkazem `for` bez výčtu položek v `in`.

Pro vytvoření cílového souboru použijte přesměrování standardního výstupu do souboru s připojením obsahu na konec. Pro vytvoření mezířádku použijte příkaz `echo`. Pro počítání pořadí souboru si zaveďte proměnnou a na vytvoření jejího obsahu použijte aritmetickou expanzi.

Pro odstranění případných adresářů z cesty a ponechání jen jména souboru použijte příkaz:

```
basename adresar/adresar/soubor.txt
```

a např. příkazovou substituci.
Pro předání obsahu souboru na standardní výstup použijte příkaz `cat`.

Požádejte kliknutím o ověření splnění úkolu.

```
#!/bin/bash

JMENO=${1}
POCET=1
shift
for PARAM; do
    echo "-- ${POCET}. `basename $PARAM` c[redacted]E" >> $JMENO
    cat $PARAM >> $JMENO
    POCET=$(($POCET + 1))
done
```

47. Funkce v bashi (fce)

Úkol:

Klasickým ukázkovým příkladem pro vysvětlení funkce v programovacích jazycích bývá použití funkce pro rekurzivní výpočet faktoriálu. Ve funkci v bashi, kvůli omezeným možnostem funkcí, jde vcelku o výzvu. Proto použijeme na výpočet faktoriálu libovolně přesný kalkulátor **bc** (vizte dřívější úkoly bc1 a bc2). A aby IS mohl řešení zkontrolovat, vložíte definici funkce do skriptu.

Napište skript:

```
/home/      /pv004lab/ukol_fce_     .sh
```

který v sobě definuje shellovou funkci nazvanou `factorial`. Potom ve skriptu použijte volání funkce `factorial` pro výpočet faktoriálu čísla zadaného prvním argumentem skriptu. Vypočtenou hodnotu předejte na standardní výstup.
Níže uvádím kód, který lze uvnitř funkce `factorial` použít na výpočet faktoriálu pomocí **bc**. Kdo si však chce výpočet v **bc** vymyslet sám (není to složité), nechť nekliká na:

Výpočet faktoriálu pomocí **bc**.

Dále ve skriptu ošetřete, abyste funkci `factorial` nevolali, pokud je první argument skriptu nezadaný, nečíselný, záporný nebo by prvním argumentem bylo číslo, které by způsobilo výpočtem faktoriálu přetečení rozsahu zobrazení 64bitové aritmetiky ve dvojkovém doplňkovém kódu. V takovém případě ukončete skript s návratovým kódem 1 bez jakéhokoli hlášení na výstupy.

Dovnitř skriptu uveďte šikanózní poznámku

```
# autor 7
```

Požádejte kliknutím o ověření splnění úkolu.

```
#!/bin/bashrc
if test $# -eq 0 || ! [ $1 -eq $1 ] 2>/dev/null || test $1 -lt 0 || test $1 -ge 21 2> /dev/null; then exit 1; fi
factorial(){ echo "define f (x) { if (x <= 1) return (1); return (f(x-1) * x); }; f($1)" | bc -l;}
factorial $1
# autor 7[redacted]
```

48. Skript ve vnořeném a aktuálním shellu, exportování proměnných (sae)

Úkol: Nejprve si kliknutím zde připravte potřebný adresář a skripty na Aise.

Nastavte si adresář `/home/ /pv004lab` jako pracovní.

V pracovním adresáři vytvořte skript

```
spust_  .sh
```

ve kterém provedete tyto operace:

1. Zavedete proměnnou `DIRc65` a nastavíte jí hodnotu `/home/: 'pv004lab/ukol_sae_` , tj. absolutní cestu k adresáři se skripty, které vytvořil IS.
2. Zavedete proměnnou `RODIC_PID` a nastavíte jí jako hodnotu číslo procesu aktuálního shellu. Tj. hodnotu, na kterou se expanduje zvláštní proměnná $$.
3. Proměnná `RODIC_PID` musí být označena k exportování (dědění) do vnořených shellů, proměnná `DIRc65` ne.
4. Ze skriptu spusťte skript `$DIRc65/vnoreny_ .sh` ve vnořeném shellu (skript `$DIRc65/vnoreny_ .sh` jste vytvořili kliknutím na "připravte...").
5. Dále ze skriptu spusťte skript `$DIRc65/aktualni_ .sh` v aktuálním shellu (skript `$DIRc65/aktualni_ .sh` jste vytvořili kliknutím na "připravte...").
6. Skript ukončete.

Spusťte skript `spust_ .sh` a zkontrolujte hlášení informující o tom, jak jsou proměnné nastaveny.
Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Skript ve vnořeném a aktuálním shellu, exportování proměnných (sae)

```
#!/bin/bash
DIRc65=/home/[redacted]/pv004lab/ukol_sae_[redacted]
export RODIC_PID=$$
$DIRc65/vnoreny_[redacted].sh
source $DIRc65/aktualni_[redacted].sh
exit 0
```

49. Příkaz getopts na zpracování voleb (geo)

Napište v bashi skript `ukol_geo_ l.sh`, který bude akceptovat tyto volby:

- **-h**
  (help) Vypíše na standardní chybový výstup nápovědu ke spuštění.
- **-a** "*Libovolně dlouhý text*"
  (append) Na konec datového souboru přidá další řádek textu zadaného argumentem.
- **-d**
  (delete) Smaže datový soubor.
- **-o** "*Libovolně dlouhý text*"
  (overwrite) Přepíše datový soubor řádkem textu zadaného argumentem.
- **-v**
  (view) Vypíše na standardní výstup celý obsah datového souboru.

Skript nechť je v adresáři `/home/..../pv004lab`. Data ve skriptu ukládejte do souboru `/home/ /pv004lab/ukol_geo_ .data`.
Absolutní cestu k souboru s daty vložte do proměnné `FILE`. Na zápisy do souboru s daty použijte operace přesměrování výstupu (typicky z příkazu `echo`).

Skript nechť se chová dle této ukázky:

```
./ukol_geo_ .sh
Žádná volba, použijte -h pro nápovědu (vypisuje se na stderr a nastavuje návratový kód na 1).
./ukol_geo_ .sh -h
Skript se spouští s volbami (vypisuje se na stderr):
-h (help) vypíše nápovědu
-a (append) přidá další data
-d (delete) smaže data
-o (overwrite) přepíše data souboru argumentem
-v (view) předá na standardní výstup obsah dat
./ukol_geo_ .sh -o "První data"
./ukol_geo_ .sh -a "Druhá data" -a "Třetí data" -v
První data
Druhá data
Třetí data
./ukol_geo_ .sh -o "Jiná data"
./ukol_geo_ .sh -v
Jiná data
./ukol_geo_ .sh -d
./ukol_geo_ .sh -v
./ukol_geo_ .sh
Žádná volba, použijte -h pro nápovědu (vypisuje se na stderr a nastavuje návratový kód na 1).
./ukol_geo_ .sh -n
./ukol_geo_ .sh: chybný přepínač – n (tento řádek vypisuje getopts na stderr)
Chybná volba, použijte -h pro nápovědu (vypisuje se na stderr).
```

Volby **-d** a **-v** nechť nevypisují chybu na stderr (standardní chybový výstup), pokud datový soubor neexistuje.
Obsah textů vypisovaných na stderr (help, chybná volba, žádná volba) se nekontroluje. Texty však musí být neprázdné právě tehdy, když mají být vypsány.

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Příkaz getopts na zpracování voleb (geo)

```
#!/bin/bash
FILE=/home/[redacted]/pv004lab/ukol_geo_[redacted].data
if test "$#" -eq 0;
        then echo "Žádná volba, použijte -h pro nápovědu" >&2; exit 1; fi
while getopts 'ha:do:v' VOLBA; do
case "$VOLBA" in
h)      echo "Skript se spouští s volbami" >&2;
        echo "-h (help) vypíše nápovědu" >&2;
        echo "-a (append) přidá další data" >&2;
        echo "-d (delete) smaže data" >&2;
        echo "-o (overwrite) přepíše data souboru argumentem" >&2;
        echo "-v (view) předá na standardní výstup obsah dat" >&2;;
a)      echo $OPTARG >> $FILE;;
d)      rm $FILE 2>/dev/null;;
o)      echo $OPTARG > $FILE;;
v)      cat $FILE 2>/dev/null;;
?)      echo "Chybná volba, použijte -h pro nápovědu" >&2; exit 1;;
esac
done
```

53. Stream editor sed, pro úpravy obsahu standardního výstupu (se1)

Úkol:
Napište skript `/home/:  /pv004lab/ukol_se1_  .sh`, který upraví výstup příkazu **who**:

```
xferda99 pts/641      2022-04-18 15:03 (99.99.broadband2.iol.cz)
xmravene pts/561      2022-02-25 17:42
```

do podoby:

```
Úkol e807a: Uživatel xferda99 z adresy 99.99.broadband2.iol.cz
Úkol e807a: Uživatel xmravene
```

Tzn. že ve skriptu spustíte příkaz **who**, jeho standardní výstup kolonou propojíte na standardní vstup příkazu **sed**, kterým přidáte úvodní řetězec každého řádku '`Úkol  a: `' a upravíte výstup příkazu **who** do požadované podoby.
Výstup seřaďte dle abecedy a odstraňte opakující se řádky (na odstranění opakujících se řádků použijte příkaz **uniq**).

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Stream editor sed, pro úpravy obsahu standardního výstupu (se1)

```
#!/bin/bash
(who|sed -e 's/^\([^ ]*\) [^(]*$/Úkol e[redacted]a: Uživatel \1/' -e 's/^\([^ ]*\) .*(\(.*\))/Úkol e[redacted]a: Uživatel \1 z adresy \2/') > /home/[redacted]/pv004lab/test.txt
sort test.txt > test2.txt
uniq test2.txt > test3.txt
cat test3.txt
```

54. Stream editor sed, pro modifikace obsahu proměnných (se2)

Úkol:
Shell při spouštění příkazu, programu, skriptu, když nezadáte cestu, prohledává adresáře dle obsahu proměnné **PATH**. Vizte:

```
echo $PATH
```

Cílem je vytvořit proměnnou **MYPATH** naplněnou obsahem proměnné **PATH** s následující změnou: **každý** řetězec znaků '`/usr/`' bude nahrazen řetězcem '`/9  l/`'.
Aby však bylo možné Vaše řešení zkontrolovat, postupujte následovně:

Napište skript `/home/  /pv004lab/ukol_se2_  .sh`, který převezme obsah prvního pozičního parametru. Spouštějte skript např. následovně:

```
./ukol_se2_  .sh $PATH
```

Skript nechť dle vzoru výše vytvoří a naplní proměnnou `MYPATH`, do které vloží dle zadání modifikovaný obsah prvního pozičního parametru (nikoli tedy přímo proměnné `PATH`).
Posledním příkazem skriptu nechť je

```
echo $MYPATH
```

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu Stream editor sed, pro modifikace obsahu proměnných (se2)

57. Stream editor sed, pro změnu obsahu v koloně (se5)

Úkol: Nejprve si kliknutím zde připravte potřebný soubor na Aise.

Vytvořil se soubor `/home/ /pv004lab/formula1_circuits.csv`

Nahlédněte na strukturu souboru výpisem jeho prvních 10 řádků:

```
head formula1_circuits.csv
```

Napište skript `/home/ /pv004lab/ukol_se5_ .sh`, který se bude spouštět s argumentem `formula1_circuits.csv`, tj.:

```
./ukol_se5_ .sh formula1_circuits.csv
```

Obsahem skriptu nechť je:

```
cat $1 | sed ...
```

Parametry příkazu **sed** zkonstruujte tak, aby **sed** četl standardní vstup a tento upravil do podoby (vizte úvodní řádky souboru `formula1_circuits.csv`):

```
In Australia, there is the Albert Park Grand Prix Circuit in Melbourne (number 114).
In Malaysia, there is the Sepang International Circuit in Kuala Lumpur (number 214).
In Bahrain, there is the Bahrain International Circuit in Sakhir (number 314).
...
```

Pozn.: Skript je zde použit jen proto, aby bylo možné jednoduše zkontrolovat použité parametry spuštění sedu.

Požádejte kliknutím o ověření splnění úkolu.

```
#!/bin/bash
cat $1 | sed '1,$s/^\([^;]\+\);\([^;]\+\);\([^;]\+\);\([^;]\+\);\([^;]\+\);\([^;]\+\).*$/In \6, there is the \4 in \5 (number \1)./'
```

60. fgrep, vyhledávání vzorku bez aplikace reg. výrazů (gr3)

Úkol:
Napište skript `/home/ /pv004lab/ukol_gr3_ .sh`, který vypíše na standardní chybový výstup chybovou zprávu a skončí s návratovým kódem 1, pokud některý z pozičních parametrů obsahuje znak svislá čára (|).

Otestujte příkazem **fgrep** a na jeho standardní vstup vložte obsah zvláštní proměnné **$***, která vždy obsahuje všechny poziční parametry.

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu fgrep, vyhledávání vzorku bez aplikace reg. výrazů (gr3)

```
#!/bin/bash
if echo $* | fgrep -q "|"; then echo xxx >&2 && exit 1; fi
```

61. cut, výběr sloupců v souboru (cut)

Úkol:
Napište skript `/home/ /pv004lab/ukol_cut_ .sh`, který vypíše na standardní výstup seznam všech dvojic login a UserID ze souboru `/etc/passwd` (sloupce 1 a 3). Sloupce oddělte dvojtečkou (:).

Dále nechť skript skončí návratovým kódem `0`, pokud je číslo UserID, zadané při spouštění skriptu prvním argumentem, nalezeno ve 3. sloupci `/etc/passwd`. Pokud není nalezeno, nebo není zadáno, skript nechť skončí návratovým kódem `1`. Příklad spuštění:

```
/home/ /pv004lab/ukol_cut_ .sh 3
```

Otestujte příkazem **fgrep** a na jeho standardní vstup vložte obsah zvláštní proměnné **$***, která vždy obsahuje všechny poziční parametry.

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu cut, výběr sloupců v souboru (cut)

```
#!/bin/bash
cat /etc/passwd | cut -f1,3 -d:
cat /etc/passwd | cut -f3 -d: > t.txt
if test $# -lt 1;then exit 1; fi
for i in $*; do if fgrep -qx $i t.txt; then exit 0; fi; done
rm t.txt
exit 1
```

62. printf, formátování výstupu (prf)

Úkol:
Napište skript /home/ /pv004lab/ukol_prf_ .sh, který se volá s jedním nepovinným argumentem. Argumentem je regulární výraz, kterým vyberete řádky souboru /etc/passwd. Vybrané řádky vypište ve tvaru:

```
          games | 12    | 100
            ftp | 14    | 50
         nobody | 99    | 99
  avahi-autoipd | 170   | 170
```

Nutno dodržet formátování (login na 16 znaků zarovnaný doprava, oddělovač ' | ', UID na 5 znaků zarovnané doleva, oddělovač ' | ', GID a nový řádek). Příklady spuštění:

```
/home/  /pv004lab/ukol_prf_  .sh ss
/home/  /pv004lab/ukol_prf_  .sh
```

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu printf, formátování výstupu (prf)

```
#!/bin/bash
if test $# -eq 1;
        then fgrep $1 /etc/passwd | (IFS=:; while read LOG PAS CUZ GID INFO HOME SHELL ZBYTEK;
                                                do printf '%16s | %-5s | %s \n' $LOG $CUZ $GID ; done);
        else cat /etc/passwd|(IFS=:; while read LOG PAS CUZ GID INFO HOME SHELL ZBYTEK;
                                                do printf '%16s | %-5s | %s \n' $LOG $CUZ $GID ; done); fi
```

63. find, průchod adresářovým stromem (fi1)

Úkol: Nejprve si kliknutím zde připravte potřebný adresář se soubory na Aise.

Napište skript /home/ /pv004lab/ukol_fi1_ .sh, který na standardní výstup předá číslo odpovídající počtu **obyčejných souborů** v adresářovém stromu /home/ /pv004lab/ukol_fi1_ , které jsou větší než **608 bajtů** a současně nemají ve jméně souborů podřetězec e 1.

Použijte příkaz **find**. Pro posouzení velikosti použijte `-size 608c`

Požádejte kliknutím o ověření splnění úkolu.

- Ověřit správnost vyřešení úkolu find, průchod adresářovým stromem (fi1)

```
#!/bin/bash
find /home/[redacted]/pv004lab/ukol_fi1_[redacted] -type f -size +608c ! -name '*e[redacted]1*'| wc -l
```