Metropolia

UNIX/Linux -ohjelmointiympäristö

HARJOITUKSIA 1

Kevät 2021 / Auvo Häkkinen

Tee tehtävät **edunix.metropolia.fi**-palvelimella (saat yhteyden sinne PuTTY:llä). Voit käyttää haluamaasi editoria - edunixissä on käytettävissä esimerkiksi nano ja emacs. Koodin tulee kääntyä **gcc**-kääntäjällä. Lisänä voi käyttää esim. valitsimia -std=c11 -pedantic -Wall.

Omat ratkaisut tulee palauttaa OMA:han tehtävien antamista seuraavan viikon aikana. Myöhemmin palautetuista saa vain puolet tehtävärukseista.

1. KOMENTORIVIARGUMENTIT

Tee C-kielellä ohjelma, joka tulostaa komentoriviargumenttien lukumäärän sekä komentoriviargumentit (mielivaltainen määrä) käänteisessä järjestyksessä. Ohjelma huomauttaa, jos sille ei anneta yhtään komentoriviargumenttia. Esimerkki

```
$ nano teht1.c                                  editoi
$ gcc -std=c11 -pedantic -Wall teht1.c          käännä
$ ./a.out                                       suorita
Ohjelman nimi on ./a.out
Et antanut komentorivillä yhtään argumenttia
$ ./a.out eka 2
Ohjelman nimi on ./a.out
Annoit ohjelman nimen perässä 2 komentoriviargumenttia
argv[2] = 2
argv[1] = eka
argv[0] = a.out
```

Huomaa: Jos työhakemistosi ei ole PATH-asetuksessa, on ohjelmaa käynnistettäessä annettava sen koko hakupolku. Piste viittaa nykyiseen työhakemistoon.

2. YMPÄRISTÖMUUTTUJA tai SITTEN KAIKKI

Kirjoita ohjelma, jolle annetaan yksi komentoriviargumentti. Jos ympäristömuuttujissa on argumenttina annettu muuttuja (esim. PATH), ohjelma tulostaa vain ko. ympäristömuuttujan arvon, esim.

```
$ ./a.out PATH
PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:
/home/hakka/Uloy/Ratk
```

Jos argumenttina annetun nimistä ympäristömuuttujaa ei ole, ohjelma asettaa ensin ko. ympäristömuuttujan arvoksi "EI_OLE_ASETETTU", sekä tulostaa vielä kaikkien ympäristömuuttujien arvot.

```
$ ./a.out KURSSI
HOSTNAME=edunix.metropolia.fi
SHELL=/bin/bash
... jne..
KURSSI=EI OLE ASETETTU
```

Huomaa: putenv() ei muuta main()-metodin saaman envp[]-taulukon sisältöä, mutta muuttaa muuttujan *environ arvoa. Muista myös, että äitiprosessi ei näe lapsiprosessin tekemiä ympäristömuuttujien muutoksia.

Vihje: muotoillun merkkijonon voi tuottaa (”katenoida kasaan”) näppärästi funktiolla sprintf()

3. KOMENTORIVILLÄ OLEVAT VALITSIMET - getopt()

Tutustu kirjastofunktion getopt(3) käyttöön luentomateriaalin, manuaalisivun ja mahdollisesti www:stä kaivamasi tekstien avulla.

Ohjelmoi yksinkertainen komentorivivalitsimin toimiva nelilaskin. Rivillä annetaan yksi arvo (argumentti) ja siihen kohdistetaan valitsimin rivillä annetut laskutoimitukset. Valitsimet ovat

`-a n`	lisää (add) argumenttiin luku n
`-s n`	vähennä (substract) argumentista luku n
`-d n`	jaa (divide) argumentti luvulla n
`-m n`	kerro (multiply) argumentti luvulla n

Laskutoimitukset tehdään esiintymisjärjestyksessä; operaattoreiden presedenssisääntöjä ei tarvitse tässä ensimmäisessä karvalakkimallissa miettiä. Esimerkkejä

```
$ gcc -Wall teht2.c -o calc    // objektitiedoston nimeksi calc
$ calc
Usage: calc argument_value [-a n] [-s n] [-d n] [-m n]
$ calc 100
100
$ calc 100 -a 10 -d 2  // argumentti 100, lasketaan (100 + 10) / 2
55
$ calc 4 -a 3 -m 3     // argumentti 4, lasketaan (4 + 3) * 3
21
$ calc 50 -d 0
Not allowed to divide by zero
```

Vihje: Siirrä aluksi varsinainen argumentti viimeiseksi toistamalla getopt()-kutsua. Poimi argumentti kohdasta optarg. Aseta sitten optind takaisin alkuarvoonsa (1), ja käsittele uudessa toistossa valitsimet ja laskutoimitukset yksi kerrallaan. Merkkijonon voi muuntaa kokonaisluvuksi funktiolla atoi().

4. KOMENTOJEN SUORITUTTAJA

Advanced Programming in the UNIX Environment kirjan (3 ed.) sivulla 12 (OMA-työtilassa on linkki kirjan PDF-tiedostoon) on annettu yksinkertaisen "komentojen suorituttaja" -ohjelman koodi. Sen kautta voi suorituttaa muita ohjelmia.

```
#include <unistd.h>  /* POSIX API */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#define MAXLINE 256

int main(void) {
  char buf[MAXLINE];
  pid_t pid;
  int status;

  printf("%% ");      /* tulosta kehoite: %%, jotta tulostuu % */
  while (fgets(buf, MAXLINE, stdin) != NULL) {

    buf[strlen(buf) - 1] = '\0'; /* korvaa rivinvaihtomerkki */

    if ((pid = fork()) < 0) {
        perror("fork error");
        exit(EXIT_FAILURE);
    }

    if (pid == 0) { /* lapsiprosessi jatkaa tästä */
        execlp(buf, buf, (char *)0);
```

```
            fprintf(stderr,"couldn't execute execlp: %s - %s",
                            buf, strerror(errno));
            exit(EXIT_FAILURE);
        }

        /* mammaprosessi jatkaa tästä */
        if ((pid = waitpid(pid, &status, 0)) < 0) {
            perror("waitpid error");
            exit(EXIT_FAILURE);
        }
        printf("%% ");
      }
      exit(EXIT_SUCCESS);
    }
```

a) Tutustu ylläolevaan koodiin ja lue systeemikutsun fork(2) ja kirjastorutiinin execlp(3) manuaalisivut. Kokeile ohjelman toimintaa: anna syötteeksi esim. ls tai pwd tai whoami (tai joku muu UNIX-komento, joka ei tarvitse komentoriviargumentteja). Syötteiden loppuminen (eli EOF, end-of-file) ilmaistaan UNIXissa näppäilemällä Ctrl-D.

5. KOMENTOJEN SUORITUTTAJA

b) Muuta edellistä koodia siten, että se välittää käynnistämälleen komennolle myös argumentteja. Nykyisessä muodossahan ohjelma osaa suorituttaa esim. komennon `ls`, mutta ei esimerkiksi komentoa `ls -l -a H1-1.c`

Vihje: man 3 exec (valitset niistä varmaankin execvp():n), merkkijonosta saa erotettua osia funktiolla strtok() tai sscanf(); Lue toki huolella manuaalisivut!