Η ΓΛΩΣΣΑ C

Μάθημα 24:

Χειρισμός Λαθών

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Θεωρία

1. Χειρισμός Λαθών
    1. Λάθη και διαχείριση λαθών
    2. Η προκαθορισμένη ροή λαθών stderr
    3. Έξοδος από το πρόγραμμα
    4. Η μεταβλητή errno
    5. Η συνάρτηση strerror
    6. Η συνάρτηση perror

Β. Ασκήσεις

Α. Πίνακες

1. Χειρισμός Λαθών

1. Λάθη και Διαχείριση τους

- Μέρος κάθε (μεγάλου) προγράμματος είναι η διαχείριση των λαθών που γίνονται κατά την εκτέλεση του προγράμματος
- Τα λάθη αυτά ποικίλουν:
  - Λάθος στο άνοιγμα ενός αρχείου
  - Σφάλμα εισόδου/εξόδου
  - Άρνηση λόγω δικαιωμάτων (permission denied)

- Για τη διαχείριση των λαθών η C προσφέρει κάποιους μηχανισμούς:
  - Τη ροή stderr
  - Την καθολική μεταβλητή errno
  - Τις συναρτήσεις strerror/perror

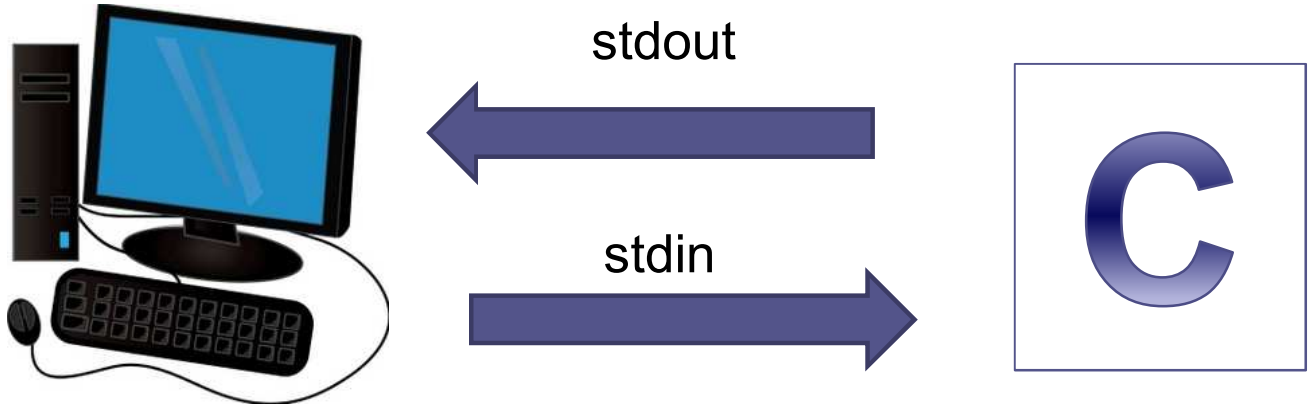
- Ο μηχανισμός της C δεν είναι και ο καλύτερος που υπάρχει σε σχέση με άλλες γλώσσες προγραμματισμού
  - Στις επόμενες γενιές γλωσσών (C++, Java) θα βρούμε πιο αποδοτικούς τρόπους διαχείρισης λαθών, όπως π.χ. ο χειρισμός εξαιρέσεων.

Α. Πίνακες

1. Χειρισμός Λαθών

2. Η προκαθορισμένη ροή λαθών stderr

- Στο μάθημα 15 «Συναρτήσεις εισόδου» κάναμε μία συζήτηση για τις προκαθορισμένες ροές:
  - εισόδου (stdin)
  - εξόδου (stdout)



- και αναφέραμε και την προκαθισμένη ροή εξόδου λαθών:
  - Η **stderr** (standard error) η οποία
    - έχει οριστεί ειδικά για να μην μπλέκεται η έξοδος (stdout)
    - με τα λάθη (stderr)
  - και by default βγάζει την έξοδο της στην οθόνη.

Α. Πίνακες

1. Χειρισμός Λαθών

2. Η προκαθορισμένη ροή λαθών stderr

- Συνεπώς μια πιο συνεπής χρήση του μηνύματος που βγάζαμε όταν διαπιστώναμε κάποιο λάθος (π.χ. αδυναμίας δέσμευσης μνήμης) είναι αντί να χρησιμοποιούμε μια απλή εκτύπωση στην οθόνη, π.χ.:

```
if (!p)
{
  printf("Adynamia desmeusis mnimis");
  exit(0);
}
```

- να στέλνουμε το μήνυμα στη ροή stderr χρησιμοποιώντας την fprintf:

```
if (!p)
{
  fprintf(stderr, "Adynamia desmeusis mnimis");
  exit(0);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

2. Η προκαθορισμένη ροή λαθών stderr

- Η stderr έχει by default έξοδο στην οθόνη.
  - Άρα οι δύο εντολές δεν θα έχουν κάποια άμεση διαφορά.
- Ωστόσο μπορούμε να ανακατευθύνουμε την έξοδο της stderr σε αρχείο
  - Εκτελώντας (από την γραμμή εντολής) με όρισμα στο πρόγραμμα το

```
2>filename.txt
```

  - Το 2> κωδικοποιεί ότι θα κάνουμε ανακατεύθυνση εξόδου στην stderr
  - και το filename.txt είναι ένα όνομα της αρεσκείας μας στο οποίο θα τυπωθούν τα μηνύματα λάθους.

Α. Πίνακες
1. Χειρισμός Λαθών
2. Η προκαθορισμένη ροή λαθών stderr

- Βλέπουμε και ένα παράδειγμα:

```
/* stderr.c */
#include <stdio.h>
#include <stdlib.h>

main()
{
   int *p;

   printf("Enarksi programmatos...\n");

   p=(int *)malloc(sizeof(int)*100000000000);
   if (!p)
   {
       fprintf(stderr, "Adynamia Desmeusis mnimis!\n");
       exit(0);
   }
   printf("Telos programmatos\n");
   free(p);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

2. Η προκαθορισμένη ροή λαθών stderr

- Εκτελώντας το πρόγραμμα από τη γραμμή εντολής:

```
stderror.exe
```

  - και η stdout και η stderr τυπώνονται στην οθόνη.

- Εκτελώντας το πρόγραμμα από τη γραμμή εντολής:

```
stderror.exe 2>error.txt
```

  - Η stdout τυπώνεται στην οθόνη
  - Η stderr τυπώνεται στο αρχείο error.txt

- Εκτελώντας το πρόγραμμα από τη γραμμή εντολής:

```
stderror.exe >output.txt 2>error.txt
```

  - Η stdout τυπώνεται στο αρχείο output.txt
  - Η stderr τυπώνεται στο αρχείο error.txt

Α. Πίνακες

1. Χειρισμός Λαθών

3. Έξοδος από το Πρόγραμμα

- Έχουμε ήδη δει τη συνάρτηση exit, η οποία τερματίζει το πρόγραμμα.
- Στην πραγματικότητα η συνάρτηση αυτή μπορεί να χρησιμοποιηθεί για να επιστρέψουμε περισσότερες πληροφορίες στο κέλυφος που εκτελεί το πρόγραμμά μας.
- Συγκεκριμένα στο stdlib.h έχουν οριστεί δύο συμβολικές σταθερές, που μπορούν να περιγράψουν την κατάσταση εξόδου.
- Αυτές είναι:

```
EXIT_SUCCESS
```

  - Στα περισσότερα συστήματα, η τιμή της είναι 0
- και

```
EXIT_FAILURE
```

  - Στα περισσότερα συστήματα (αλλά όχι πάντα) η τιμή της είναι 1
  - και ενδέχεται να κωδικοποιεί ακόμη και τον τύπο του λάθους.

- Λόγω των διαφορετικών τιμών που έχουν αυτές οι δύο σταθερές ανάλογα με το μεταγλωττιστή μας, οι γνώμες διίστανται για το κατά πόσον πρέπει να τις χρησιμοποιούμε στα προγράμματά μας.
  - Αν πάντως ξέρουμε το σύστημα που θα τρέξει το πρόγραμμα μας, τότε θα μπορούμε να τις χρησιμοποιούμε.

Α. Πίνακες

1. Χειρισμός Λαθών

3. Έξοδος από το Πρόγραμμα

- Βλέπουμε και την τροποποιήση του προγράμματός μας με βάση τις σταθερές που ορίσαμε:

```
/* stderr2.c */
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int *p;

   printf("Enarksi programmatos...\n");

   p=(int *)malloc(sizeof(int)*100000000000);
   if (!p)
   {
       fprintf(stderr, "Adynamia Desmeusis mnimis!\n");
       exit(EXIT_FAILURE);
   }
   printf("Telos programmatos\n");
   free(p);
   exit(EXIT_SUCCESS);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

3. Έξοδος από το Πρόγραμμα

- Η έξοδος επιστρέφεται στην διεργασία που κάλεσε το πρόγραμμά μας (π.χ. από την κονσόλα)
- Υπάρχει η δυνατότητα να πάρουμε αυτήν την τιμή μέσω εντολών συστήματος
  - Αλλά ξεφεύγει από τα όρια αυτού του μαθήματος
  - Και είναι μέρος scripting γλωσσών που προσφέρει το λειτουργικό σύστημα.

- Επίσης να αναφέρουμε ότι:
  - Η exit κάνει κάποιες βασικές λειτουργίες πριν τερματίσει το πρόγραμμα:
    - Εκτυπώνει δεδομένα που δεν έχουν γίνει εγγραφή σε ροές εξόδου
    - Κλείνει τα ανοικτά αρχεία
    - Επιστρέφει τον ακέραιο που είναι το όρισμα της στο λειτουργικό σύστημα.

Α. Πίνακες

1. Χειρισμός Λαθών

4. Η μεταβλητή errno

- Οι περισσότερες συναρτήσεις βιβλιοθήκης, σε περίπτωση λάθους, αναθέτουν μία τιμή σε μία καθολική μεταβλητή (την errno)
  - Η τιμή αυτή χρησιμοποιείται για να ανιχνευτεί το λάθος και να υπάρχει μια πιο φιλική αντιμετώπιση στο κλείσιμο του προγράμματος.
- Για να χρησιμοποιήσουμε την errno, πρέπει να ενσωματώσουμε την κεφαλίδα:

```
errno.h
```

- Προσοχή!
  - Κάθε κλήση συνάρτησης μπορεί να επηρεάσει την errno
  - Γι'αυτό πρέπει να αξιοποιήσουμε την τιμή της αμέσως μετά την κλήση που προκάλεσε το σφάλμα.

- Βλέπουμε και ένα παράδειγμα:

Α. Πίνακες

1. Χειρισμός Λαθών

4. Η μεταβλητή errno

- Βλέπουμε και την τροποποιήση του προγράμματός μας με βάση τις σταθερές που ορίσαμε:

```
/* errno.c */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
int main()
{
   int *p;

   printf("Enarksi programmatos...\n");

   p=(int *)malloc(sizeof(int)*100000000000);
   if (!p)
   {
      fprintf(stderr, "errno: %d\n", errno);
      fprintf(stderr, "Adynamia Desmeusis mnimis!\n");
      exit(EXIT_FAILURE);
   }
   printf("Telos programmatos\n");
   free(p);
   exit(EXIT_SUCCESS);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

4. Η μεταβλητή errno

- Στο πρόγραμμα της προηγούμενης διαφάνειας, θα εκτυπωθεί:

```
Enarksi programmatos...
errno: 12
Adynamia Desmeusis mnimis!
```

- Οι κωδικοί της errno έχουν οριστεί εσωτερικά να αντιστοιχούν σε προβλήματα που μπορεί να προκύψουν κατά την εκτέλεση του προγράμματος.
- π.χ. στο https://gist.github.com/greggyNapalm/2413028
  - Αναφέρονται 130 περίπου τέτοιες τιμές, και διαβάζουμε μεταξύ άλλων:

```
#define EPERM 1 /* Operation not permitted */
#define ENOENT 2 /* No such file or directory */
...
#define ENOMEM 12 /* Out of memory */
```

  - Συνεπώς το λάθος αναγνωρίζεται ως «out of memory» (εξαντλήθηκε η μνήμη)

Α. Πίνακες

1. Χειρισμός Λαθών

5. Η συνάρτηση strerror

- Η συνάρτηση strerror:

```
char *strerror(int errnum)
```

  - Παίρνει σαν όρισμα κάποιο errno και επιστρέφει ένα string με μία εύληπτη περιγραφή μηνύματος λάθους.
- Π.χ. η κλήση:

```
strerror(12)
```

  - Θα επιστρέψει το μήνυμα «Out of Memory»

Α. Πίνακες

1. Χειρισμός Λαθών

5. Η συνάρτηση strerror

- Και μια εφαρμογή:

```
/* strerror.c */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main()
{
   int *p;

   printf("Enarksi programmatos...\n");

   p=(int *)malloc(sizeof(int)*100000000000);
   if (!p)
   {
      fprintf(stderr, "ERROR %d:%s\n", errno, strerror(errno));
      exit(EXIT_FAILURE);
   }
   printf("Telos programmatos\n");
   free(p);
   exit(EXIT_SUCCESS);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

6. Η συνάρτηση perror

- Η συνάρτηση perror:

```
void perror(char *str)
```

  - Τυπώνει:
    - Τη συμβολοσειρά str, ακολουθούμενη από ανω κάτω τελεία
      - ή τίποτα αν βάλουμε ως όρισμα NULL
    - Και στη συνέχεια το μήνυμα που αντιστοιχεί στο λάθος που συνέβη τελευταίο (και η τιμή του είναι στο errno).
- Π.χ. η κλήση:

```
perror("Error: ");
```

  - Θα επιστρέψει το μήνυμα «Error: Out of Memory»

Α. Πίνακες

1. Χειρισμός Λαθών

6. Η συνάρτηση perror

- Και μια εφαρμογή:

```
/* perror.c */
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>

int main()
{
   int *p;

   printf("Enarksi programmatos...\n");

   p=(int *)malloc(sizeof(int)*100000000000);
   if (!p)
   {
       perror(0);
       exit(EXIT_FAILURE);
   }
   printf("Telos programmatos\n");
   free(p);
   exit(EXIT_SUCCESS);
}
```

Α. Πίνακες

1. Χειρισμός Λαθών

7. Περαιτέρω μελέτη

- Ένα πρόγραμμα λειτουργεί κάτω από το λειτουργικό σύστημα
- Συνεπώς όταν λέμε ότι επιστρέφουμε μία τιμή από το πρόγραμμα, αυτή πηγαίνει στο λειτουργικό σύστημα.
- Υπάρχουν τρόποι το λειτουργικό σύστημα να αξιοποιήσει αυτήν την τιμή,
  - …αλλά αυτό ξεφεύγει από τα όρια της συγκεκριμένης σειράς μαθημάτων!

- Περισσότερα μαθαίνουμε όταν μελετάμε προγραμματισμό συστήματος, σε κάποιο λειτουργικό, π.χ. στο linux…

Β. Ασκήσεις

1. Πείραμα: Λάθος στο άνοιγμα αρχείου

Γράψτε ένα πρόγραμμα το οποίο:

1. Να ανοίγει ένα αρχείο το οποίο δεν υπάρχει.
2. Εκτυπώστε τον κωδικό λάθους και το μήνυμα λάθους με την perror