Προγραμματισμός Ι

Προχωρημένα Θέματα

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Ανακατεύθυνση Εισόδου/Εξόδου

Συνήθως η τυπική είσοδος ενός προγράμματος (`stdin`) προέρχεται από το πληκτρολόγιο. Αντίστοιχα η τυπική έξοδος (`stdout`) και η τυπική έξοδος λάθους (`stderr`) είναι συνήθως η οθόνη.

Τα διάφορα λειτουργικά συστήματα μας επιτρέπουν να ανακατευθύνουμε την είσοδο και την έξοδο.

Μπορούμε για παράδειγμα να ανακατευθύνουμε την τυπική έξοδο σε ένα αρχείο ή να χρησιμοποιήσουμε ένα αρχείο ως τυπική είσοδο. Οι λεπτομέρειες εξαρτώνται από το λειτουργικό σύστημα.

Ανακατεύθυνση Εισόδου

Έστω ένα πρόγραμμα `sum` που διαβάζει από την τυπική είσοδο ακεραίους μέχρι να συναντήσει `EOF` και τυπώνει στην τυπική έξοδο το άθροισμα των αριθμών.

Αντί να δώσουμε είσοδο στον πρόγραμμα μέσω του πληκτρολογίου μπορούμε να φτιάξουμε ένα αρχείο `input` στο οποίο να αποθηκεύσουμε τους αριθμούς. Στην συνέχεια μπορούμε να εκτελέσουμε το πρόγραμμα ως:

```
$ sum < input
```

Το παραπάνω παράδειγμα δείχνει την εντολή σε ένα περιβάλλον Unix (σε συστήματα Unix το $ είναι η προτροπή γραμμής εντολής).

Ανακατεύθυνση Εισόδου

```
$ sum < input
```

Σε περιβάλλον DOS (ή στα σύγχρονα windows όπου με cmd.exe προσομοιώνουμε περιβάλλον γραμμής εντολών DOS) η ανακατεύθυνηση εισόδου λειτουργεί με τον ίδιο τρόπο.

Διοχέτευση

piping

Διοχέτευση είναι ανακατεύθυνση της τυπικής εξόδου ενός προγράμματος στην τυπική είσοδο ενός άλλου προγράμματος.

Αν υποθέσουμε πως το πρόγραμμα `random` παράγει μια σειρά τυχαίων αριθμών, τότε γράφοντας

```
$ random | sum
```

μπορούμε να υπολογίσουμε το άθροισμα αυτών των αριθμών. Η διοχέτευση μπορεί να συνεχιστεί και σε τρίτο πρόγραμμα.

Ανακατεύθυνση Εξόδου

Για να ανακατευθύνουμε την τυπική έξοδο προς ένα αρχείο με όνομα `output` μπορούμε να γράψουμε (είτε σε περιβάλλον Unix είτε DOS)

```
$ sum > output
```

Τώρα το πρόγραμμα ζητάει τους αριθμούς από το πληκτρολόγιο και θα γράψει την έξοδο του (το άθροισμα) στον αρχείο output. Το μέγεθος του αρχείου αυτού μηδενίζεται πριν την χρήση, άμα υπάρχει ήδη.

Προσάρτηση Εξόδου

Για να προσαρτηθεί η έξοδος στο τέλος ενός ήδη υπάρχοντος αρχείου μπορούμε να χρησιμοποιήσουμε το σύμβολο προσάρτησης εξόδου (>>).

```
$ sum >> output
```

Πολλαπλή Χρήση

Μπορούμε να χρησιμοποιήσουμε τις ανακατευθύνσεις ταυτόχρονα. Έστω το αρχείο **input**

```
2
6
1
3
```

Τότε χρησιμοποιώντας τα προγράμματα **cat** (τυπώνει ένα αρχείο) και **sort** (ταξινομεί την είσοδο)

```
$ cat input | sort -n > output
```

πέρνουμε το αρχείο **output**

```
1
2
3
6
```

Λίστες Ορισμάτων Μεταβαλλόμενου Μήκους

Η συνάρτηση `printf()` που χρησιμοποιούμε τόσο καιρό μπορεί να πάρει ένα μεταβαλλόμενο μήκος παραμέτρων.

π.χ

```
printf("Hello world!\n");
```

ή

```
printf("i = %d !\n", i);
```

ή

```
printf("i = %d, f = %f !\n", i, f);
```

Λίστες Ορισμάτων Μεταβαλλόμενου Μήκους

Η συνάρτηση `printf()` ορίζεται ως εξής:

```
int printf( const char *format, ... );
```

Τα αποσιωπητικά (...) δηλώνουν ότι η συνάρτηση δέχεται ένα μεταβαλλόμενο πλήθος ορισμάτων οποιαδήποτε τύπου.

Τα αποσιωπητικά πρέπει να τοποθετούνται πάντοτε στο τέλος της λίστας παραμέτρων. Πρέπει να υπάρχει τουλάχιστον μια παράμετρος που δεν είναι αποσιωπητικά (...) .

Λίστες Ορισμάτων Μεταβαλλόμενου Μήκους

`stdarg.h`

Οι μακροεντολές και οι ορισμοί του αρχείου επικεφαλίδας `stdarg.h` παρέχουν τις δυνατότητες που απαιτούνται για την δόμηση συναρτήσεων με λίστες ορισμάτων μεταβαλλόμενου μήκους.

Τύποι

- `va_list` - τύπος για πρόσβαση στα ορίσματα

Μακροεντολές

- `va_start` - αρχίζει την πρόσβαση στις παραμέτρους
- `va_arg` - διαβάζει την τρέχουσα παράμετρο και προχωράει στο επόμενο
- `va_end` - ελευθερώνει ένα `va_list`
- `va_copy` - αντιγραφή δεδομένων ενός `va_list` σε άλλο

Λίστες Ορισμάτων Μεταβαλλόμενου Μήκους

Παράδειγμα

```c
#include <stdio.h>
#include <stdarg.h>

double average(int i, ...) {
    double total = 0.0;
    int j;

    va_list ap;
    va_start(ap, i);

    for(j = 1; j <= i; j++)
        total += va_arg(ap, double);

    va_end(ap);
    return total / i;
}

main() {
    double w = 1.0, x = 2.0, y = 3.0;
    printf("%lf\n", average(3, w, x, y));
}
```

Τερματισμός Προγράμματος με `exit` και `atexit`

Μέσω της βιβλιοθήκης γενικών βοηθημάτων (`stdlib.h`) μπορούμε να τερματίσουμε ένα πρόγραμμα, με διαφορετικό τρόπο από την συμβατική επιστροφή από την συνάρτηση `main`.

Η συνάρτηση `exit`

- συνήθως χρησιμοποιείται για να τερματίσει ένα πρόγραμμα όταν ανιχνευτεί ένα σφάλμα, π.χ στην είσοδο ή αν ένα απαραίτητο αρχείο δεν μπορεί να ανοιχθεί.

Τερματισμός Προγράμματος με `exit` και `atexit`

Μέσω της βιβλιοθήκης γενικών βοηθημάτων (`stdlib.h`) μπορούμε να τερματίσουμε ένα πρόγραμμα, με διαφορετικό τρόπο από την συμβατική επιστροφή από την συνάρτηση `main`.

Η συνάρτηση `atexit`

- δεσμεύει μια συνάρτηση στο πρόγραμμα να καλείται με τον επιτυχή τερματισμό του -- δηλ., είτε όταν το πρόγραμμα τερματίζεται φτάνοντας στο τέλος της `main`, είτε όταν εμπλακεί η `exit`.

Τερματισμός Προγράμματος με `exit` και `atexit`

Η συνάρτηση `atexit` φαίνεται παρακάτω:

```
int atexit (void (*FUNCTION) (void));
```

Η παράμετρος είναι μια συνάρτηση χωρίς παραμέτρους. Η συνάρτηση `atexit` επιστρέφει 0 εαν επιτύχει και διάφορο του μηδενός σε περίπτωση αποτυχίας.

Οι συναρτήσεις που δηλώνονται μέσω της συνάρτησης `atexit` καλούνται σε αντίστροφη σειρά από εκείνη της δέσμευσης τους.

Τερματισμός Προγράμματος με `exit` και `atexit`

```
void exit(int STATUS);
```

Πέρνει παραμέτρους τις δύο ακέραιες σταθερές:

- **`EXIT_SUCCESS`**
- **`EXIT_FAILURE`**

Παράδειγμα Τερματισμού

```c
#include <stdio.h>
#include <stdlib.h>

void bye(void) {
    puts("Goodbye....");
}

int main(void) {
    atexit(bye);
    exit(EXIT_SUCCESS);
}
```

Η δεσμευμένη λέξη `register` λέει στον μεταγλωττιστή να αποθηκεύσει μια μεταβλητή σε ένα καταχωρητή (CPU register) εάν είναι δυνατό.

```
1  #include <stdio.h>
2
3  main() {
4      register int i;
5
6      for(i = 0; i < 100; i++)
7          printf("%4d", i);
8  }
```

Ο λόγος χρήσης της δεσμευμένης λέξης `register` είναι η βελτιστοποίηση της ταχύτητας αφού οι καταχωρητές του επεξεργαστή είναι πιο κοντά στον επεξεργαστή και άρα ταχύτεροι από την μνήμη.

Η δεσμευμένη λέξη `register` λέει στον μεταγλωττιστή να αποθηκεύσει μια μεταβλητή σε ένα καταχωρητή (CPU register) εάν είναι δυνατό.

Ο λόγος χρήσης της δεσμευμένης λέξης `register` είναι η βελτιστοποίηση της ταχύτητας αφού οι καταχωρητές του επεξεργαστή είναι πιο κοντά στον επεξεργαστή και άρα ταχύτεροι από την μνήμη.

Οι σύγχρονοι μεταγλωττιστές τρέχουν πολύπλοκους αλγορίθμους και υπολογίζουν καλύτερα από τον άνθρωπο ποιες μεταβλητές πρέπει να είναι σε καταχωρητές. Καλύτερα λοιπόν να μην χρησιμοποιούμε καθόλου αυτή την δυνατότητα της γλώσσας.

Το προσδιοριστικό τύπου `volatile` είναι ουσιαστικά το αντίθετο από το προσδιοριστικό τύπου `const`.

```
#include <stdio.h>

main() {
    volatile int i;

    /* code here */
}
```

Δηλώνει πως μια μεταβλητή μπορεί να αλλάξει με τελείως απρόβλεπτο τρόπο (για παράδειγμα μέσω interrupts ή μέσω κάποιου παράλληλου προγράμματος, κ.τ.λ). Κάθε χρήση της μεταβλητής θα ξαναδιαβάσει τα δεδομένα από την μνήμη αντί να χρησιμοποιήσει για παράδειγμα ένα αντίγραφο της μεταβλητής από κάποιον καταχωρητή.