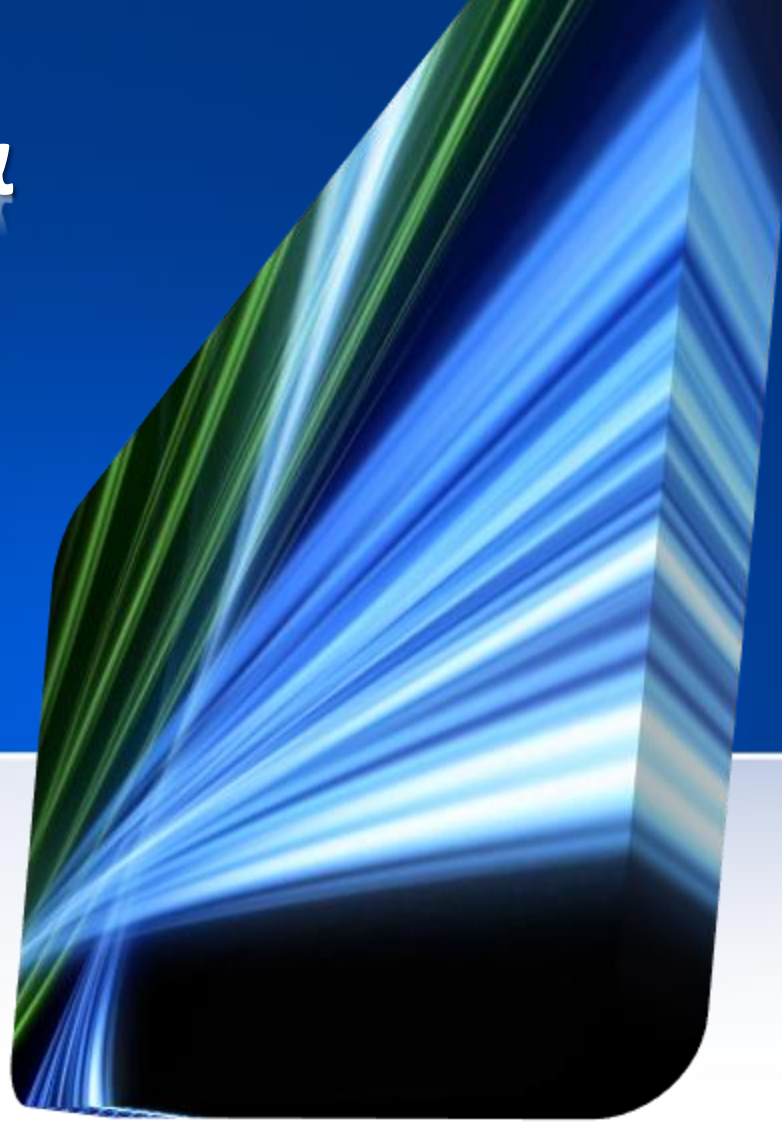
Λειτουργικά Συστήματα
6ο εξάμηνο ΣΗΜΜΥ
Ακ. έτος 2023-2024



Εργαστηριακή Άσκηση 3

Χειρισμός Διοχετεύσεων

Εργαστηριακή Άσκηση 3

Να γραφτεί πρόγραμμα σε γλώσσα προγραμματισμού C και περιβάλλον Linux στο οποίο η διεργασία πατέρας δημιουργεί **n** διεργασίες παιδία και τους μοιράζει μέσω διοχετεύσεων (pipes) εργασίες που της αναθέτει ο χρήστης από την γραμμή εντολών.

Εργαστηριακή Άσκηση 3

Οι διεργασίες επικοινωνούν και συγχρονίζονται μέσω διοχετεύσεων (pipes)

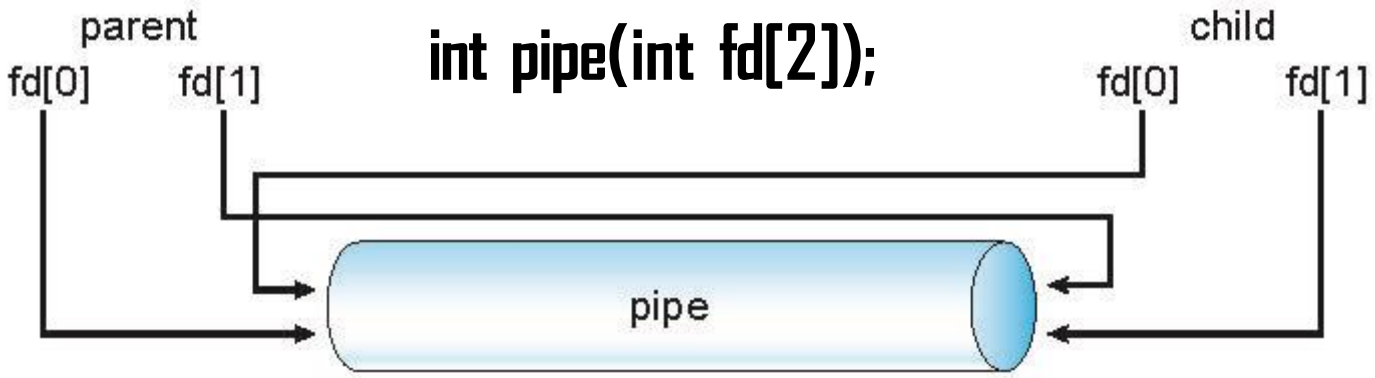
Διοχετεύσεις Pipes (θεωρία)

- Λειτουργούν ως σωληνώσεις που επιτρέπουν σε δύο διεργασίες να επικοινωνούν
- Οι συνηθισμένες διοχετεύσεις δεν είναι προσβάσιμες έξω από την διεργασία που την δημιούργησε.
- Συνήθως, μια γονική διαδικασία δημιουργεί μια διοχέτευση και την χρησιμοποιεί για να επικοινωνεί με μια διεργασία που δημιούργησε.
- Επώνυμες διοχετεύσεις (Named pipes) – είναι προσβάσιμες χωρίς να υπάρχει σχέση γονέα-παιδιού ή παιδιών ίδιου πατέρα.

Κοινές Διοχετεύσεις
Ordinary Pipes

- Οι κοινές διοχετεύσεις επιτρέπουν την επικοινωνία σε τυπικό τρόπο παραγωγού-καταναλωτή
- Ο παραγωγός γράφει στο ένα άκρο (άκρο εγγραφής **write-end**)
- Ο καταναλωτής διαβάζει από το άλλο άκρο (άκρο ανάγνωσης **read-end**)
- Επομένως, οι κοινές διοχετεύσεις είναι μονής κατεύθυνσης
- Απαιτείται σχέση γονέα-παιδιού μεταξύ των διεργασιών επικοινωνίας
- Στα Windows καλούνται ως ανώνυμες διοχετεύσεις **anonymous pipes**

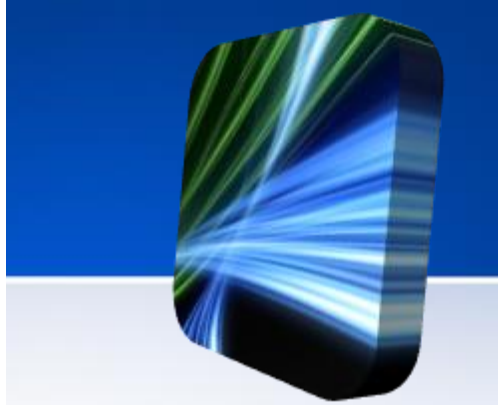
Κοινές Διοχετεύσεις
Ordinary Pipes



Η κλήση συστήματος **pipe()** δημιουργεί μια διοχέτευση, ένα κανάλι δεδομένων μονής κατεύθυνσης που μπορεί να χρησιμοποιηθεί για επικοινωνία μεταξύ διεργασιών. Ο πίνακας fd χρησιμοποιείται για την επιστροφή δύο περιγραφητών αρχείων που αναφέρονται στα άκρα της διοχέτευσης. Το στοιχείο fd[0] του πίνακα αναφέρεται στο άκρο ανάγνωσης της διοχέτευσης και το fd [1] αναφέρεται στο άκρο εγγραφής.

Περισσότερες πληροφορίες:
https://linux.die.net/man/2/pipe

Παράδειγμα Διοχετεύσεων



```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(void)
{
        int pd[2], nbytes;
        pid_t childpid;
        char string[] = "Hello, world!\n";
        char readbuffer[80];
        pipe(pd);

        if((childpid = fork()) == -1) {
                perror("fork");
                exit(1);
        }
        if(childpid == 0) {
                close(pd[0]);
                write(pd[1], string, (strlen(string)+1));
                exit(0);
        }
        else {
                close(pd[1]);
                nbytes = read(pd[0], readbuffer, sizeof(readbuffer));
                printf("Received string: %s", readbuffer);
        }
        return(0);
}
```

Λεπτομέρειες Άσκησης 3

Παράδειγμα εκτέλεσης προγράμματος

./ask3 n - -round-robin ή **./ask3 n - -random**

Το όρισμα **n** πρέπει να είναι ακέραιος θετικός αριθμός που προσδιορίζει το πλήθος των διεργασιών παιδιών που θα δημιουργήσει ο πατέρας.

Το όρισμα **round-robin** ή **random** προσδιορίζει τον τρόπο με τον οποίο η διεργασία πατέρας μοιράζει τις εργασίες:

round-robin: με κυκλική επαναφορά

random: με τυχαίο τρόπο

Λεπτομέρειες Άσκησης 3

Παράδειγμα εκτέλεσης προγράμματος

./ask3 n - -round-robin ή **./ask3 n - -random**

Αν δεν δοθούν σωστά τα ορίσματα η διεργασίας πατέρας εκτυπώνει το παρακάτω μήνυμα και τερματίζει

Usage: ask3 <nChildren> [--random] [--round-robin]

Ως default μεθοδολογία διαμοιρασμού εργασιών εάν δεν προσδιοριστεί από τον χρήστη θεωρείτε η **round-robin**
π.χ. εκτέλεση **./ask3 6**

Λεπτομέρειες Άσκησης 3

Η διεργασία πατέρας μπορεί να δεχθεί από τον χρήστη μέσω του terminal τις παρακάτω εντολές

- **help**. Στην περίπτωση αυτή η διεργασία πατέρας εκτυπώνει το μήνυμα: **Type a number to send job to a child!**
- **exit**. Στην περίπτωση αυτή η διεργασία πατέρας τερματίζει αφού πρώτα τερματίσει όλα τα παιδιά εκτυπώνοντας κατάλληλα μηνύματα π.χ. **All children terminated**
- Έναν ακέραιο αριθμό π.χ. **12**. Στην περίπτωση αυτή η διεργασία πατέρας διαλέγει μια διεργασία παιδί και της αναθέτει την εργασία στέλνοντας της τον αριθμό μέσω μιας διοχέτευσης και εκτυπώνει το μήνυμα **[Parent] Assigned 12 to child 5**
- Σε κάθε άλλη περίπτωση να εκτυπώνεται το μήνυμα **Type a number to send job to a child!**

Λεπτομέρειες Άσκησης 3

Κάθε διεργασία παιδί μπορεί να δεχθεί από την διεργασία πατέρα μέσω μιας διοχέτευσης ένα αριθμό τον οποίο μειώνει κατά 1, περιμένει για 10 secs και τον επιστρέφει στην διεργασία πατέρα μέσω μιας άλλης διοχέτευσης. Αυτό γίνεται επαναληπτικά μέχρι να την τερματίσει η διεργασία πατέρας

Απλό παράδειγμα χωρίς ελέγχους:

```
int val;
while(1) {
        read(in, &val, sizeof(int))
        printf("[Child] [%d] Child received %d!"\n", gIndex, getpid(), val);
        val--;
        sleep(10);
        write(out, &val, sizeof(int))
        printf("[Child %d] [%d] Child Finished hard work, writing back %d"\n", gIndex, getpid(), val);
    }
```

Όπου **in** και **out** είναι άκρα ανάγνωσης και εγγραφής σε διοχετεύσεις, και **gIndex** η αρίθμηση του παιδιού

Λεπτομέρειες Άσκησης 3

Συνεπώς η διεργασία πατέρας μπορεί να δεχθεί δεδομένα από ένα σύνολο διοχετεύσεων ή από το terminal, αυτό είναι εφικτό να γίνει μέσω της **poll()**.
Η **poll()** επιτρέπει σε ένα πρόγραμμα να παρακολουθεί πολλούς περιγραφητές αρχείων, έως ότου ένας ή περισσότεροι από τους περιγραφητές αρχείων γίνουν «έτοιμοι».

Περισσότερες πληροφορίες:

- https://man7.org/linux/man-pages/man2/poll.2.html
- αρχείο στον φάκελο *Χρήσιμα Αρχεία* της Ασκήσης 3

Λεπτομέρειες Άσκησης 3

- Για τυχαία παραγωγή αριθμών που είναι πιθανό να σας χρειαστεί στον διαμοιρασμό των εργασιών με random τρόπο προτείνεται η rand()
- Για δυναμική διάθεση μνήμης που είναι πιθανό να σας χρειαστεί για τον ορισμό πινάκων αφού διαβάσετε το **n** από την γραμμή εντολών προτείνεται η malloc()

Περισσότερες πληροφορίες:

- http://man7.org/linux/man-pages/man3/malloc.3.html
- http://man7.org/linux/man-pages/man3/rand.3.html
- https://www.tutorialspoint.com/c_standard_library/c_function_rand.htm
- https://www.tutorialspoint.com/c_standard_library/c_function_malloc.htm

Χρήσιμα links

- http://man7.org/linux/man-pages/man2/pipe.2.html
- http://www.gnu.org/software/libc/manual/html_node/Pipes-and-FIFOs.html#Pipes-and-FIFOs
- https://linux.die.net/man/2/pipe
- https://man7.org/linux/man-pages/man2/poll.2.html
- http://man7.org/linux/man-pages/man3/malloc.3.html
- http://man7.org/linux/man-pages/man3/rand.3.html
- https://www.tutorialspoint.com/c_standard_library/c_function_rand.htm
- https://www.tutorialspoint.com/c_standard_library/c_function_malloc.htm