ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

Chủ đề: Xv6- Unix Utilities

HỆ ĐIỀU HÀNH - 23CLC01

Hồ Chí Minh, ngày 05 tháng 03 năm 2025

LỜI CẢM ƠN

Đầu tiên, chúng em xin bày tỏ lòng biết ơn và gửi lời cảm ơn chân thành đến thầy Lê Quốc Hòa người đã hướng dẫn giúp chúng em hiểu được mục tiêu và mình cần phải làm gì để hoàn thành đồ án này.

Tiếp theo, chúng em xin cảm ơn các thành viên trong nhóm vì đã làm việc hết mình, hỗ trợ lẫn nhau và phối hợp hiệu quả trong suốt quá trình thực hiện đồ án này

MỤC LỤC

I. Thông tin nhóm:

1. Bảng thành viên:

MSSV	Họ và tên	Lớp
23127419	Cao Phạm Tuấn Minh	23CLC01
23127364	Đặng Nguyễn Thành Hiếu	23CLC01
23127232	Trần Hoàng Nam	23CLC01

2. Bảng phân công công việc:

STT	Yêu cầu	Người thực hiện	Mức độ hoàn thành
1	Pingpong	Đặng Nguyễn Thành Hiếu··	100%
2	Primes	Đặng Nguyễn Thành Hiếu··	100%
3	Find	Trần Hoàng Nam	100%
4	Xargs	Cao Phạm Tuấn Minh	100%
5	Báo cáo	Trần Hoàng Nam Đặng Nguyễn Thành Hiếu Cao Phạm Tuấn Minh	100%

II. Thực hành đồ án:

1. Ping Pong:

1.1. Yêu cầu:

- Viết một chương trình cấp người dùng sử dụng các lời gọi hệ thống của xv6 để truyền một byte "ping-pong" giữa hai tiến trình thông qua một cặp ống dẫn(pipe), một ống cho mỗi hướng. Tiến trình cha sẽ gửi một byte đến tiến trình con. Tiến trình con sẽ in ra: "<pid>: received ping", trong đó <pid> là ID của tiến trình con, sau đó ghi byte vào ống dẫn để gửi lại cho tiến trình cha rồi thoát. Tiến trình cha sẽ đọc byte từ tiến trình con, in ra: "<pid>: received pong", và thoát.

1.2. Cách thức hoạt động:

- Chương trình sử dụng pipe để giao tiếp giữa tiến trình cha và tiến trình con trong xv6. Tiến trình cha gửi "ping" qua pipe, tiến trình con đọc, in "received ping", rồi gửi "pong" lại. Cha nhận "pong", in "received pong", sau đó cả hai đóng pipe và kết thúc. Quá trình diễn ra tuần tự nhờ cơ chế blocking read/write của pipe.

1.3. Nguyên lý hoạt động của thuật toán:

- Tạo 2 mảng có 2 phần tử, tương ứng với hai ống dẫn:
    + p1[2]: Truyền dữ liệu từ tiến trình cha đến tiến trình con.
    + p2[2]: Truyền dữ liệu từ tiến trình con đến tiến trình cha.
- Dùng hàm pipe() để tạo hai pipe. Nếu tạo không thành công, in thông báo lỗi và thoát chương trình.

- Dùng hàm fork() để tạo tiến trình con.
    + Nếu giá trị trả về là 0, tiến trình hiện tại là tiến trình con.
    + Nếu giá trị trả về > 0, tiến trình hiện tại là tiến trình cha.
    + Sau khi fork(), hai tiến trình sẽ chạy song song.
- Nếu là tiến trình con:
    + Đóng đầu ghi của tiến trình cha (p1[1]) và đầu đọc của tiến trình con (p2[0]) để hạn chế rò rỉ dữ liệu.
    + Đọc dữ liệu từ tiến trình cha qua p1[0].
    + In ra thông báo “received ping”.
    + Ghi dữ liệu “pong” vào p2[1] để gửi lại cho tiến trình cha.
    + Đóng các đầu pipe còn lại (p1[0], p2[1]) và kết thúc tiến trình con.
- Nếu là tiến trình cha:
    + Đóng đầu ghi của tiến trình con (p2[1]) và đầu đọc của tiến trình cha (p1[0]).
    + Kiểm tra và ghi "ping" vào p1[1] để gửi sang tiến trình con.
    + Kiểm tra xem tiến trình con có gửi dữ liệu "pong" về không bằng cách đọc từ p2[0].
    + Nếu đọc thành công, in ra "received pong".
    + Đóng các đầu pipe còn lại (p1[1], p2[0]) và thoát chương trình.

2. Primes:

2.1. Yêu cầu:

- Viết một chương trình sàng số nguyên tố đồng thời cho xv6 bằng cách sử dụng pipes và thiết kế được minh họa trong bức hình ở giữa trang này cùng với phần văn bản xung quanh. Ý tưởng này là của Doug McIlroy, người phát minh ra Unix pipes. Giải pháp của bạn nên được đặt trong tệp user/primes.c.

2.2. Cách thức hoạt động:

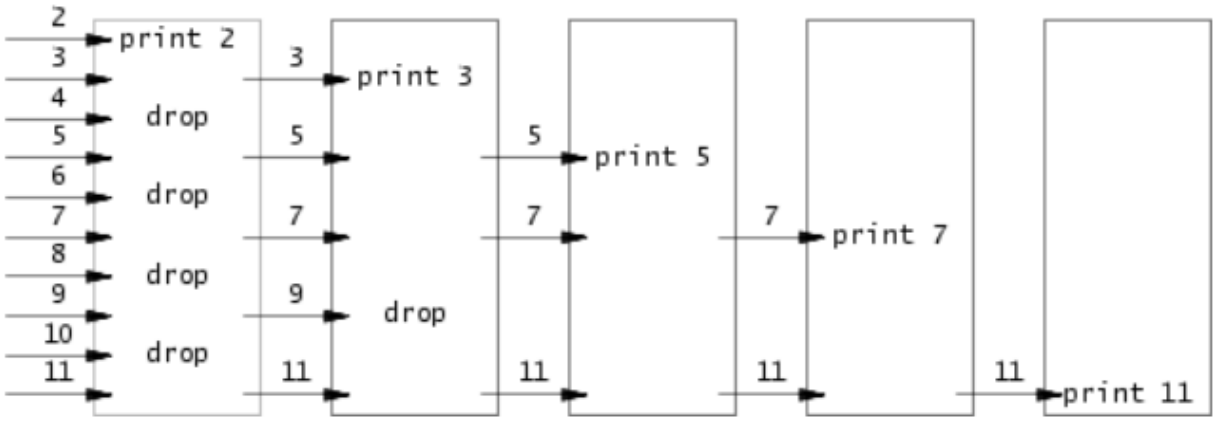
- Chương trình dùng thuật toán Sàng Eratosthenes để tìm các số nguyên tố từ 2 đến 280 thông qua cơ chế giao tiếp tiến trình bằng **pipe**. Tiến trình cha ghi dãy số từ 2 đến 280 vào pipe, sau đó tiến trình con **đọc số đầu tiên** (là số nguyên tố), in ra màn hình, rồi tạo pipe mới để chuyển tiếp các số chưa bị chia hết cho số nguyên tố đó. Quá trình này lặp lại đệ quy: mỗi tiến trình con tiếp tục đọc số đầu tiên từ pipe mới, lọc các số còn lại và truyền tiếp cho tiến trình con tiếp theo. Khi không còn số nào để xử lý, các tiến trình con kết thúc dần.

2.3. Nguyên lý hoạt động của thuật toán:

- Tiến trình cha tạo một danh sách số từ 2 đến 280 và gửi qua pipe.
- Tiến trình con đọc số đầu tiên (chắc chắn là số nguyên tố), in ra màn hình, rồi lọc bỏ các bội số của số này và truyền các số còn lại cho tiến trình con tiếp theo.
- Mỗi tiến trình con đóng vai trò như một bộ lọc, chỉ giữ lại các số không chia hết cho số nguyên tố đầu tiên của nó.
- **Bộ lọc số nguyên tố sieve(int read_fd) :**
    + Đọc số đầu tiên từ pipe (chắc chắn là số nguyên tố).
    + In ra màn hình số nguyên tố đầu tiên.
    + Tạo pipe mới (new_pipe): Dùng để truyền các số còn lại cho tiến trình con tiếp theo.
- **Tạo tiến trình con (fork()):**
    + Biến root_pid được dùng để lưu ID của tiến trình con khi tiến hành fork().
    + Nếu fork() trả về 0, nghĩa là đang trong tiến trình con.

+ Nếu fork() trả về giá trị lớn hơn 0, nghĩa là đang trong tiến trình cha và root_pid chứa PID của tiến trình con.
+ Nếu là tiến trình con, gọi đệ quy sieve(new_pipe[0]) để tiếp tục sàng.
+ Nếu là tiến trình cha, lọc bỏ các số chia hết cho số nguyên tố vừa tìm được và truyền các số còn lại qua pipe mới.

- **Hàm wait(0):**
  + Chờ tiến trình con kết thúc, tránh tạo **zombie process**.



2.4. Quản lý tài nguyên:

- Đóng pipe sau khi dùng để tránh rò rỉ tài nguyên.
- Hạn chế tạo quá nhiều tiến trình, chỉ tạo tiến tình con khi cần để tránh gây tốn tài nguyên hệ thống.

3. Find:

3.1. Yêu cầu:

- Viết một phiên bản đơn giản của chương trình tìm UNIX cho xv6: tìm tất cả các tệp trong cây thư mục có phần mở rộng tên cụ thể.

3.2. Cách thức hoạt động:

- Chương trình này là một phiên bản đơn giản của lệnh find trong hệ điều hành UNIX, được viết bằng C. Nó thực hiện tìm kiếm một tệp (file) hoặc thư mục (directory) có tên cụ thể trong một cây thư mục (directory tree) bắt đầu từ một đường dẫn gốc được chỉ định.
- Chương trình nhận hai tham số từ dòng lệnh:
    + path: Đường dẫn thư mục gốc để bắt đầu tìm kiếm.
    + filename: Tên của tệp hoặc thư mục cần tìm.
- Nếu số lượng tham số không đủ, chương trình sẽ hiển thị hướng dẫn sử dụng và thoát.

3.3. Nguyên lý hoạt động của thuật toán:

- **Hàm get_filename(char *path):**
    + Hàm này trả về phần tên của tệp (hoặc thư mục) từ một đường dẫn đầy đủ (path).
    + Cách hoạt động:
        - Bắt đầu từ cuối chuỗi path, duyệt ngược về phía trước cho đến khi gặp dấu hoặc đi hết chuỗi.
        - Trả về con trỏ trỏ tới ký tự sau dấu / (tên tệp hoặc thư mục).
- **Hàm search_file_recursive(char *dir_path, char *target_name):**
    + Kiểm tra tệp hiện tại:
        - Sử dụng hàm get_filename để lấy phần tên của đường dẫn dir_path.
        - So sánh tên này với target_name. Nếu trùng khớp, in ra đường dẫn đầy đủ của tệp hoặc thư mục.
    + Mở thư mục:

- Dùng open để mở thư mục dir_path. Nếu không mở được, thông báo lỗi và thoát.
- Dùng fstat để lấy thông tin về dir_path. Nếu dir_path không phải là thư mục (tức là không có kiểu T_DIR), thoát hàm.

+ Duyệt nội dung thư mục:
  - Sử dụng read để đọc từng mục trong thư mục.
  - Bỏ qua các mục đặc biệt "." (thư mục hiện tại) và ".." (thư mục cha).
  - Ghép tên mục hiện tại vào đường dẫn buf để tạo đường dẫn đầy đủ tới tệp hoặc thư mục con.
+ Đệ quy:
  - Gọi lại hàm search_file_recursive với đường dẫn đầy đủ vừa tạo được.
  - Điều này giúp duyệt sâu hơn vào các thư mục con.
+ Đóng tệp:
  - Sau khi duyệt xong, đóng descriptor thư mục (close(fd)).

- **Hàm main:**
  + Hàm main kiểm tra số lượng tham số đầu vào:
    - Nếu không đủ tham số, in ra hướng dẫn sử dụng và thoát.
  + Gọi hàm search_file_recursive với đường dẫn gốc (argv[1]) và tên tệp cần tìm (argv[2]).
  + Thoát chương trình sau khi hoàn thành.

4. Xargs:

4.1. Yêu cầu:

- Viết một phiên bản đơn giản của chương trình UNIX xargs cho xv6: các đối số của nó mô tả một lệnh để chạy, nó đọc các dòng từ đầu vào tiêu chuẩn và nó chạy lệnh cho mỗi dòng, nối thêm dòng đó vào các đối số của lệnh.

4.2. Cách thức hoạt động:

- Chương trình này là một phiên bản đơn giản của lệnh xargs trong UNIX, được viết bằng ngôn ngữ C. Nó đọc các dòng đầu vào (stdin), tách chúng thành các chuỗi lệnh (arguments), và thực thi một lệnh (command) được chỉ định với các đối số (arguments) đó. Chương trình hỗ trợ tùy chọn -n để chỉ định số lượng đối số tối đa sẽ được truyền vào lệnh trong một lần thực thi.

4.3. Nguyên lý hoạt động của thuật toán:

- **Xử lý tham số dòng lệnh:**
    + Chương trình bắt đầu bằng cách kiểm tra các tham số truyền vào:
        - Nếu không có đủ tham số (argc < 2), in thông báo hướng dẫn sử dụng và thoát.
        - Nếu tham số đầu tiên là -n, đọc giá trị batch_size từ tham số tiếp theo (argv[2]):
            - Nếu giá trị này không hợp lệ (<= 0), in lỗi và thoát.
            - Nếu -n được cung cấp, các tham số lệnh bắt đầu từ argv[3].
            - Nếu -n không được cung cấp, các tham số lệnh bắt đầu từ argv[1].
        - Sao chép các tham số lệnh (command và args...) vào mảng command_args.

- **Đọc từ đầu vào (stdin):**
    + Chương trình đọc từng dòng từ stdin. Mỗi dòng được lưu vào input_line và được xử lý để tách thành các đối số (tokens):
        - Loại bỏ các ký tự khoảng trắng (' ' hoặc '\t').
        - Lưu các đối số tách được vào mảng tokens.
- **Chạy lệnh với các đối số:**
    + Nếu tùy chọn -n được cung cấp (batch_size > 0):
        - Chia các đối số trong tokens thành các nhóm có kích thước tối đa là batch_size.
        - Với mỗi nhóm:
            - Tạo một tiến trình con (fork).
            - Trong tiến trình con, truyền nhóm đối số đó cùng với các tham số cố định (command_args) vào lệnh và thực thi bằng exec.
            - Tiến trình cha chờ (wait) tiến trình con kết thúc trước khi tiếp tục.
    + Nếu -n không được cung cấp:
        - Gộp tất cả các đối số từ dòng đầu vào vào command_args.
        - Tạo một tiến trình con (fork) để thực thi lệnh với tất cả các đối số.
        - Tiến trình cha chờ (wait) tiến trình con kết thúc.
- **Kết thúc chương trình:**
    + Sau khi đọc hết dữ liệu từ stdin và thực thi lệnh với tất cả các đối số, chương trình thoát.

III. Tài liệu tham khảo:

1. https://pdos.csail.mit.edu/6.1810/2024/labs/util.html
2. [File Descriptors and Open Files] https://www.youtube.com/watch?v=oWuVGDese4k
3. https://pdos.csail.mit.edu/6.1810/2024/xv6/book-riscv-rev4.pdf
4. https://github.com/whileskies/xv6-labs-2020/blob/cd99ef8666796ded0477f9b5447f3f41dec1a471/doc/Lab1-Xv6%20and%20Unix%20utilities.md
5. Chat GPT