دانشگاه علم و صنعت

دانشکده مهندسی کامپیوتر

درجه تحصیلی: کارشناسی

گزارشکار تمرین5 OS

گردآورنده:

پرنیان شاکریان - 99400064

استاد:

دکتر انتظاری

سال تحصیلی: خرداد ۱۴۰۲

خلاصه:

در تمرین پنجم سیستم عامل قصد داریم مبحث Inter-process communication (IPC) را بررسی کنیم. اگر بخواهیم به طور خلاصه توضیح دهیم، به تکنیک‌هایی اشاره دارد که توسط process برای تبادل اطلاعات، هماهنگ کردن فعالیت‌ها و همگام‌سازی عملیات آنها استفاده می‌شود. (IPC) اجازه می دهد تا processهای در حال اجرا بر روی یک سیستم با یکدیگر ارتباط برقرار کرده و مشارکت کنند.

روش های (IPC):

۱. Pipes: شکل ساده‌ای از IPC را ارائه می دهند که در آن داده از طریق یک کانال ارتباطی یک طرفه بین processها منتقل می شود. یک process، داده ها را در Pipe نوشته و process دیگر آن را می خواند.
۲. Shared Memory: به چندین process اجازه میدهد تا به بخش حافظه‌ای که در آنها مشترک است دسترسی داشته باشند. processها می توانند با mapping کردن ناحیه‌ای از حافظه مشترک در فضای آدرس خود، آن را خوانده و بنویسند و اشتراک گذاری داده ها را فراهم کنند. (مکانیسم‌های سمافور یا mutex اغلب برای هماهنگ کردن دسترسی به حافظه مشترک استفاده می‌شوند.)
۳. signals: راهی برای processها به منظور ارسال notification یا قطع سیگنال به processهای دیگر است. از سیگنال‌ها می‌توان برای مدیریت انتقال اطلاعات بین processها استفاده کرد، اما از نظر داده‌هایی که می‌توانند حمل کنند محدود هستند.

شرح کلی: برای حل سوالات تمرین در ابتدا لازم است کلیه کتابخانه‌های مورد نیاز را وارد کنیم. کلیه کتابخانه های مورد استفاده در این تمرین نمایش داده شده است.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
```

3. You are asked to use named pipes to develop a basic chat tool. The program is going to run locally twice, creating two processes that can talk with each other. They read messages from the standard input, send it to the other party, and display the received response. The username is received as a command line parameter.

ابتدای کد ثابت ها را تعریف می کنیم:

۱. MAX_BUF_SIZE حداکثر اندازه را برای بافر پیام تعیین می کند.
۲. PIPE_NAME نام pipe را مشخص می کند.

تابع main نقطه ورود برنامه است که دو آرگومان argc (argument count) و argv (بردار) را میگیرد. در داخل تابع ۴ متغیر را اعلام میکنیم:

۱. Fd: متغیر عدد صحیح است که file descriptor را برای pipe نامگذاری شده نگه میدارد.
۲. Username: آرایه ای از کاراکترها برای ذخیره نام کاربری است.
۳. Input: آرایه ای از کاراکترها برای ذخیره پیام ورودی است.
۴. Response: آرایه ای از کاراکترها برای ذخیره پاسخ دریافتی است.

در ادامه دستور if بررسی میکند که تعداد آرگومان های خط فرمان برابر با ۲ نباشد.اگر چنین بود، فرمت استفاده صحیح را چاپ و یک کد در وضعیت غیر صفر را برای نشان دادن خطا برمیگرداند. نام کاربری از آرگومان command line به آرایه username با استفاده از تابع strcpy کپی می‌شود. تابع mkfifo برای ایجاد یک pipe با نام مشخص شده توسط PIPE_NAME فراخوانی میشود. تابع fork برای ایجاد یک process فرزند فراخوانی شده و مقدار بازگشتی آن در pid ذخیره میشود. اگر pid کمتر از ۰ باشد، به این معنی است که fork شکست خورده پس یک پیغام خطا در stderr چاپ میکند و برنامه با وضعیت غیر صفر خارج میشود. در process والد که با pid > 0 مشخص شده، pipe نامگذاری شده برای نوشتن (پرچم O_WRONLY) با استفاده از تابع open باز میشود.

داخل حلقه while، از کاربر خواسته می شود پیامی را وارد کند. تابع fgets پیام را از ورودی استاندارد خوانده و آن را در آرایه ورودی ذخیره میکند. sizeof(input) تضمین میکند که فقط، حداکثر MAX_BUF_SIZE کاراکتر خوانده شود تا از سرریز بافر جلوگیری کند. پیام‌ها با استفاده از تابع write درpipe نامگذاری شده نوشته میشود. strlen(input) + 1 به عنوان آرگومان اندازه برای گنجاندن null terminator استفاده میشود. اگر کاربر "exit " را وارد کند، حلقه خاتمه می یابد. پاسخ process دیگر از pipe نامگذاری شده با استفاده از تابع read خوانده و بر روی کنسول نمایش

داده میشود. در process فرزند که با pid == 0 مشخص شده، pipe نامگذاری شده برای خواندن (O_RDONLY) با استفاده از تابع open باز میشود. در داخل حلقه while دوم، پاسخ از pipe نامگذاری شده خوانده شده و روی کنسول نمایش داده می شود. سپس از کاربر خواسته میشود تا پاسخی را وارد کند که با استفاده از تابع write در pipe نامگذاری شده نوشته می شود. پس از خروج از حلقه، process والد، pipe نامگذاری شده را با close(fd) بسته و هر دو process با برگرداندن ۰ به اجرا پایان می دهند.

برای اجرای برنامه از کامندهای زیر استفاده کنید:

```
gcc chat.c -o chat
./chat name1
./chat name2
```

ابتدا برنامه را کامپایل و اسم ها را وارد کنید.

خروجی ها: