



دانشگاه جهرم

استفاده از حافظه مشترک و سمافور در سیستم عامل لینوکس

نگارش:

امیررضا ارجمند

استاد:

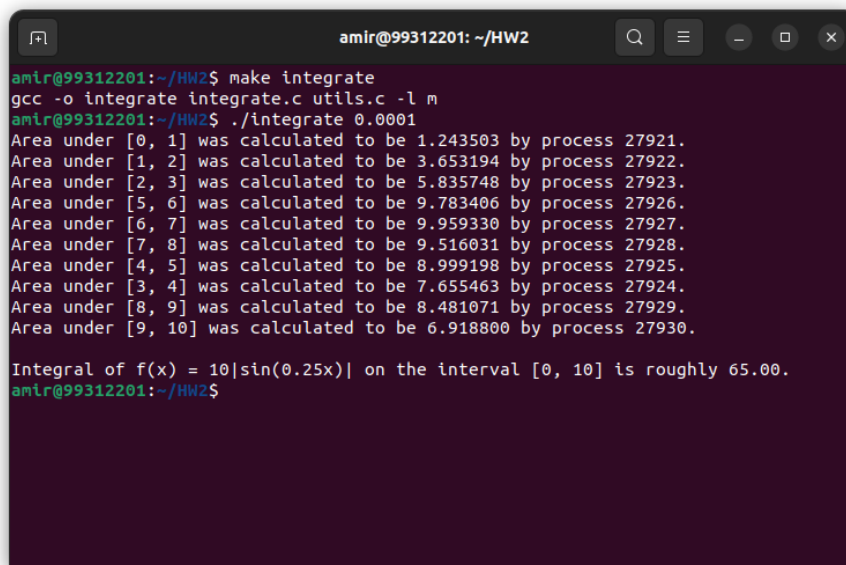
دکتر محمد جواد پارسه

شماره دانشجویی:

۹۹۳۱۲۲۰۱

۱. توضیحاتی درمورد برنامه

در این برنامه مانند گزارش کار ۲ مقدار تقریبی $\int_0^{10} 10|\sin 0.25x| dx$ را با استفاده از ۱۰ پردازش موازی و به کمک روش دوزنقه‌ای محاسبه می‌کنیم. تفاوت این برنامه در نحوه‌ی ارتباط پردازش‌ها و ارسال اطلاعات آن‌ها به یکدیگر است. در برنامه‌ی قبلی هر پردازش پس از محاسبه‌ی مقدار انتگرال در بازه‌ی اختصاص داده شده به آن، حاصل را به صورت سیگنال exit status به پردازش‌های والد ارسال می‌کرد. از آنجایی که هر پردازش فقط می‌تواند مقدار صحیحی در بازه‌ی ۰ تا ۲۵۵ را به عنوان سیگنال خروجی برای والد خود ارسال کند و انتگرال محاسبه شده توسط هر پردازش مقدار اعشاری است، قسمت اعشاری مقدار بازگشتی هر پردازش حذف می‌شد و مقدار مقدار نهایی محاسبه شده با واقعیت اختلاف قابل توجهی داشت.



```
amir@99312201: ~/HW2
amir@99312201:~/HW2$ make integrate
gcc -o integrate integrate.c utils.c -l m
amir@99312201:~/HW2$ ./integrate 0.0001
Area under [0, 1] was calculated to be 1.243503 by process 27921.
Area under [1, 2] was calculated to be 3.653194 by process 27922.
Area under [2, 3] was calculated to be 5.835748 by process 27923.
Area under [3, 4] was calculated to be 7.655463 by process 27924.
Area under [4, 5] was calculated to be 8.999198 by process 27925.
Area under [5, 6] was calculated to be 9.783406 by process 27926.
Area under [6, 7] was calculated to be 9.959330 by process 27927.
Area under [7, 8] was calculated to be 9.516031 by process 27928.
Area under [8, 9] was calculated to be 8.481071 by process 27929.
Area under [9, 10] was calculated to be 6.918800 by process 27930.

Integral of f(x) = 10|sin(0.25x)| on the interval [0, 10] is roughly 65.00.
amir@99312201:~/HW2$
```

در اینجا به جای استفاده از exit status متغیری از نوع double را میان والد و ۱۰ پردازش فرزند به اشتراک می‌گذاریم. از آنجایی که تمام پردازش‌ها قابلیت نوشتن به صورت همزمان را روی این متغیر دارند ممکن است به مشکل تناقض داده (data inconsistency) برخورد کنیم و نتیجه نهایی اشتباهی محاسبه شود. برای جلوگیری از data inconsistency، می‌توانیم از ابزاری مانند سمافور استفاده کنیم و با استفاده از آن خواندن و نوشتن را روی متغیر مشترک کنترل کنیم. بقیه‌ی جزئیات نحوه‌ی کارکرد برنامه مانند آرگمان خط دستور dx، مشابه با برنامه‌ی قبلی در گزارش کار ۲ می‌باشد.

```
amir@99312201: ~/HW4
make
gcc -o integrate integrate.c utils.c -l m -pthread
amir@99312201: ~/HW4$ ./integrate 0.0001
Area under [0, 1] was calculated to be 1.24 by process 36648. Parent process: 36647.
Area under [1, 2] was calculated to be 3.65 by process 36649. Parent process: 36647.
Area under [2, 3] was calculated to be 5.84 by process 36650. Parent process: 36647.
Area under [3, 4] was calculated to be 7.66 by process 36651. Parent process: 36647.
Area under [4, 5] was calculated to be 9.00 by process 36652. Parent process: 36647.
Area under [5, 6] was calculated to be 9.78 by process 36653. Parent process: 36647.
Area under [6, 7] was calculated to be 9.96 by process 36654. Parent process: 36647.
Area under [7, 8] was calculated to be 9.52 by process 36655. Parent process: 36647.
Area under [8, 9] was calculated to be 8.48 by process 36656. Parent process: 36647.
Area under [9, 10] was calculated to be 6.92 by process 36657. Parent process: 36647.
Integral of  $f(x) = 10|\sin(0.25x)|$  on the interval [0, 10] is roughly 72.045744618132005.
amir@99312201: ~/HW4$
```

علاوه بر این در گزارش کار خواسته شده که از تابع سیستمی `vfork` نیز استفاده شود. یکی از تفاوت‌های `fork` با `vfork` این است که `address space` مربوط به والد با فرزندان به اشتراک گذاشته می‌شود. در این صورت نیازی به ساختن متغیر مشترک نیست چون فضای حافظه والد بین تمام فرزندان مشترک است! برای مثال می‌توانیم یک آرایه با ۱۰ اندیس در والد تعریف کنیم و هر پرده جمع می‌زنند و مقدار نهایی را بدست می‌آورد. توجه داشته باشید که یکی دیگر از خصوصیات `vfork` این است که پرده والد تا اتمام اجرای فرزند صبر می‌کند و سپس به اجرا شدن ادامه می‌دهد. به همین علت است که تمام فرزندان به ترتیب فراخوانی اجرا می‌شوند و خروجی را نمایش می‌دهند. این قضیه ممکن است در سرعت کلی اجرای برنامه تاثیر منفی داشته باشد.

۲. جزئیات نحوه‌ی تولید حافظه‌ی مشترک برای برنامه

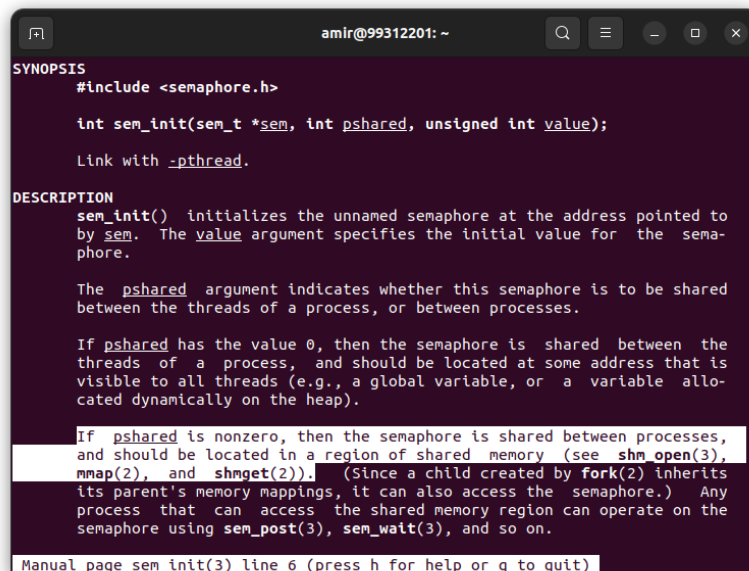
روش‌های متعددی برای تولید حافظه‌ی مشترک بین پرده‌ها وجود دارد که ما در اینجا از تابع سیستمی `mmap` استفاده می‌کنیم. کاربرد اصلی تابع `mmap` نگاشت کردن یک فایل در دیسک به بخشی از فضای مموری سیستم می‌باشد، به طوری که با دادن مقدار عددی `file descriptor` تابع به بخش مشخص کردن آرگمان‌های ورودی آدرس آن فایل را در `virtual address space` به ما برمی‌گرداند و می‌توانیم با آن مانند ساختمان داده‌ای در مموری رفتار کنیم. اما در اینجا ما فقط بخشی از فضای حافظه را می‌خواهیم و نیازی به نگاشت کردن آن با فایل خاصی نداریم. برای این کار از مکروری `MAP_ANONYMOUS` استفاده می‌کنیم و مقدار `fd` و `offset` که دو آرگمان آخر تابع می‌باشند را برابر با -۱ و ۰ قرار می‌دهیم. همچنین برای به اشتراک گذاشتن این بخش از حافظه بین پرده‌های مختلف از فلگ `MAP_SHARED` استفاده می‌کنیم. آدرس بازگشتی از `mmap` فضای حافظه‌ای است که در اختیار داریم. کد استفاده شده برای بدست آوردن آدرس یک متغیر مشترک در حافظه به شرح زیر می‌باشد.

```
double *partial_integral = mmap(NULL, sizeof(double), PROT_WRITE | PROT_READ,
MAP_SHARED | MAP_ANONYMOUS, -1, 0);
```

۳. جزئیات استفاده از سمافورها

در سیستم‌های مبتنی بر استاندارد POSIX می‌توانیم از دو نوع سمافور `named` و `unnamed` استفاده کنیم. سمافورهای `named` دارای نام مشخصی هستند و در شاخه‌ی `/dev/shm` در قالب فایل قابل مشاهده هستند. برای ساخت این سمافورها از تابع `sem_open` استفاده می‌کنیم. درمقابل این نوع سمافورها از سمافورهای `unnamed` استفاده می‌شود که دیگر به صورت فایل در شاخه‌ی ذکر شده قابل مشاهده نیست و برای ایجاد آن نیاز به حافظه مشترک بین پردازنده‌ها داریم. این نوع از سمافورها با استفاده از تابع `sem_init` ایجاد می‌شوند.

طبق `man page` مربوط به تابع `sem_init`، برای سمافور ابتدا باید به اندازه‌ی نوع داده `sem_t` فضای مشترک بین پردازنده‌ها به وجود بیاوریم و آدرس فضای مشترک را به همراه مقدار اولیه سمافور به این تابع بدهیم. سپس می‌توانیم با استفاده از توابع `sem_wait` و `sem_post` از سمافور استفاده کنیم و ویژگی انحصار متقابل را رعایت کنیم.



```
amir@99312201: ~
SYNOPSIS
#include <semaphore.h>

int sem_init(sem_t *sem, int pshared, unsigned int value);

Link with -pthread.

DESCRIPTION
sem_init() initializes the unnamed semaphore at the address pointed to
by sem. The value argument specifies the initial value for the sema-
phore.

The pshared argument indicates whether this semaphore is to be shared
between the threads of a process, or between processes.

If pshared has the value 0, then the semaphore is shared between the
threads of a process, and should be located at some address that is
visible to all threads (e.g., a global variable, or a variable allo-
cated dynamically on the heap).

If pshared is nonzero, then the semaphore is shared between processes,
and should be located in a region of shared memory (see shm_open(3),
mmap(2), and shmat(2)). (Since a child created by fork(2) inherits
its parent's memory mappings, it can also access the semaphore.) Any
process that can access the shared memory region can operate on the
semaphore using sem_post(3), sem_wait(3), and so on.

Manual page sem_init(3) line 6 (press h for help or q to quit)
```

۴. جزئیاتی راجع به اجرای برنامه با vfork

با استفاده از `vfork` به طور پیشفرض فضای مموری والد توسط فرزند قابل دسترسی می‌باشد و نیازی به ایجاد متغیر مشترک نیست. اینجا بجای استفاده از یک متغیر از یک آرایه با ۱۰ المان استفاده می‌کنیم. هر پردازنده پس از محاسبه‌ی انتگرال بخشی که به آن اختصاص شده مقدار محاسبه شده را در یکی از خانه‌های آرایه قرار می‌دهد و اندیس آن خانه را به والد به عنوان سیگنال `exit status` برمی‌گرداند. فرایند والد سپس تمامی خانه‌های آرایه را با هم جمع زده و خروجی را به کاربر نمایش می‌دهد.

۵. اجرا کردن و نمایش خروجی دو برنامه

با استفاده از دستور make می‌توانیم برنامه‌ها را کامپایل کنیم. سپس با نوشتن نام برنامه و مشخص کردن مقدار dx پاسخ انتگرال در خروجی چاپ می‌شود. توجه داشته باشید که این پاسخ بسیار دقیق‌تر از خروجی برنامه‌ی نوشته شده در گزارش کار ۲ می‌باشد.

```
amir@99312201: ~/HW4
amir@99312201:~/HW4$ make
gcc -o integrate integrate.c utils.c -l m -pthread
gcc -o integrate_vfork integrate_vfork.c utils.c -l m -pthread
amir@99312201:~/HW4$ ./integrate_vfork 0.0001
Area under [0, 1] was calculated to be 1.24 by process 2891. Parent process: 2890.
Area under [1, 2] was calculated to be 3.65 by process 2892. Parent process: 2890.
Area under [2, 3] was calculated to be 5.84 by process 2893. Parent process: 2890.
Area under [3, 4] was calculated to be 7.66 by process 2894. Parent process: 2890.
Area under [4, 5] was calculated to be 9.00 by process 2895. Parent process: 2890.
Area under [5, 6] was calculated to be 9.78 by process 2896. Parent process: 2890.
Area under [6, 7] was calculated to be 9.96 by process 2897. Parent process: 2890.
Area under [7, 8] was calculated to be 9.52 by process 2898. Parent process: 2890.
Area under [8, 9] was calculated to be 8.48 by process 2899. Parent process: 2890.
Area under [9, 10] was calculated to be 6.92 by process 2900. Parent process: 2890.

Integral of  $f(x) = 10|\sin(0.25x)|$  on the interval [0, 10] is roughly 72.0457446181320
05.
amir@99312201:~/HW4$
```

```
amir@99312201: ~/HW4
amir@99312201:~/HW4$ make
gcc -o integrate integrate.c utils.c -l m -pthread
gcc -o integrate_vfork integrate_vfork.c utils.c -l m -pthread
amir@99312201:~/HW4$ ./integrate 0.0001
Area under [0, 1] was calculated to be 1.24 by process 2728. Parent process: 2727.
Area under [8, 9] was calculated to be 8.48 by process 2736. Parent process: 2727.
Area under [2, 3] was calculated to be 5.84 by process 2730. Parent process: 2727.
Area under [3, 4] was calculated to be 7.66 by process 2731. Parent process: 2727.
Area under [4, 5] was calculated to be 9.00 by process 2732. Parent process: 2727.
Area under [9, 10] was calculated to be 6.92 by process 2737. Parent process: 2727.
Area under [6, 7] was calculated to be 9.96 by process 2734. Parent process: 2727.
Area under [7, 8] was calculated to be 9.52 by process 2735. Parent process: 2727.
Area under [1, 2] was calculated to be 3.65 by process 2729. Parent process: 2727.
Area under [5, 6] was calculated to be 9.78 by process 2733. Parent process: 2727.

Integral of  $f(x) = 10|\sin(0.25x)|$  on the interval [0, 10] is roughly 72.0457446181320
19.
amir@99312201:~/HW4$
```