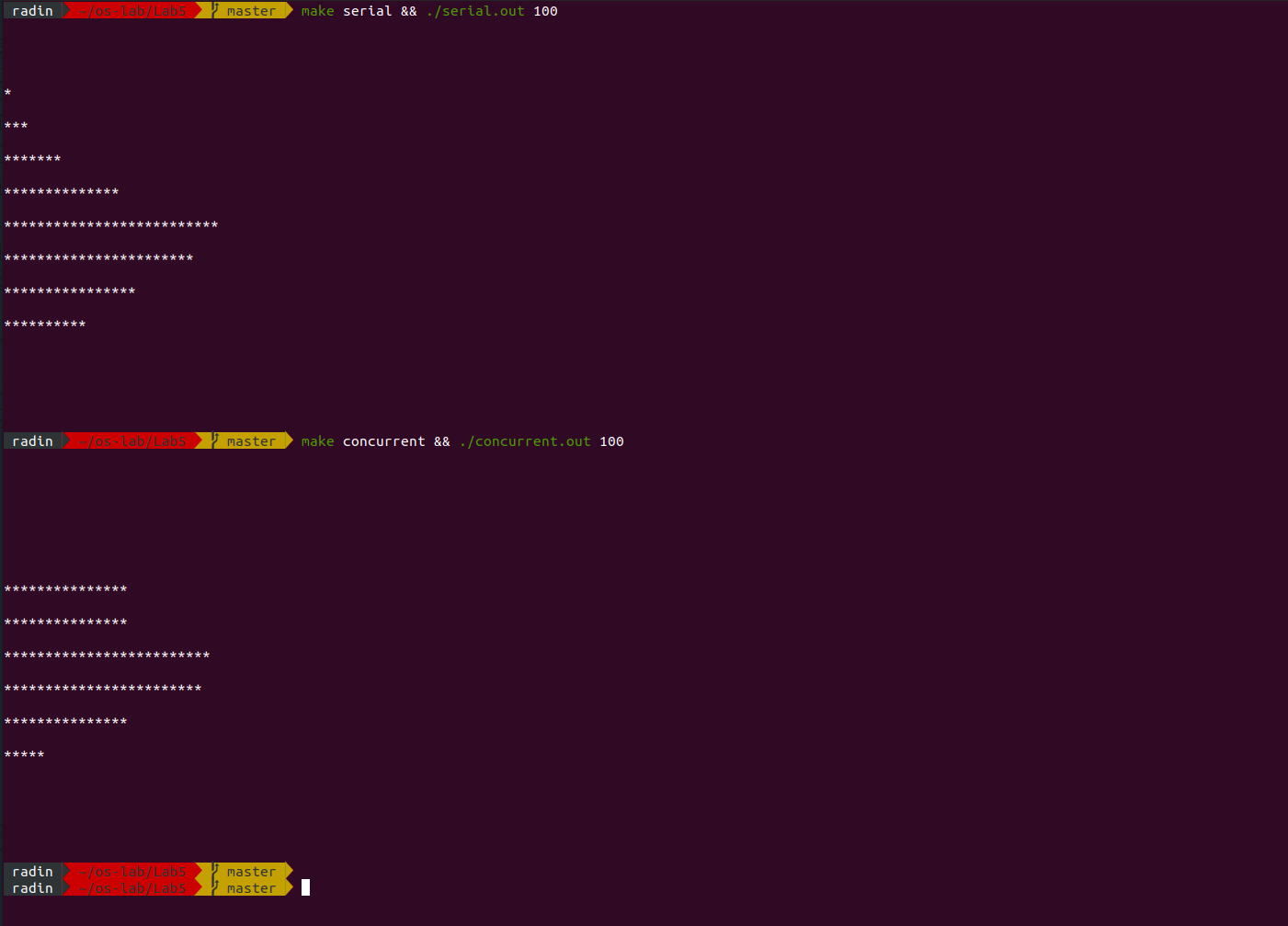


آزمایشگاه سیستم‌های عامل

رادین شایانفر

پاییز ۱۳۹۹

ابتدا با رسم هیستوگرام‌ در ۲ حالت سریال و هم‌روند به ازای نمونه‌ی ۱۰۰ تایی، درستی کد را بررسی می‌کنیم.

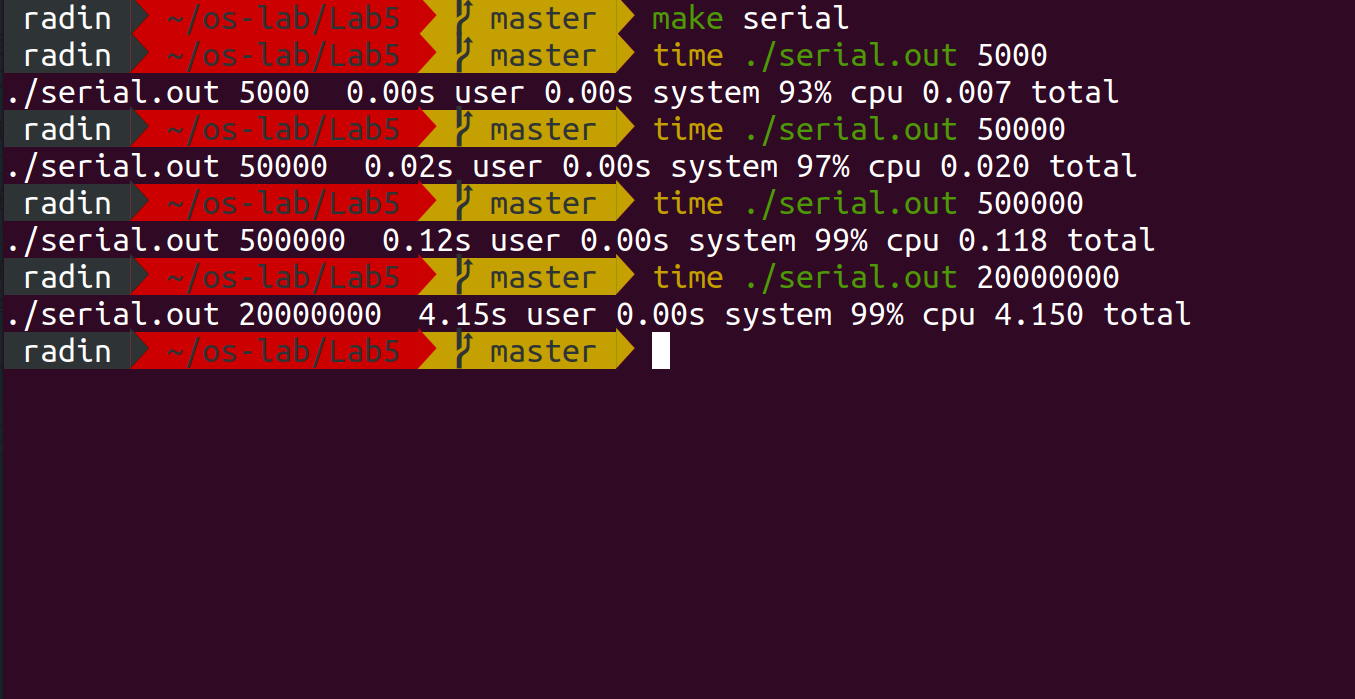
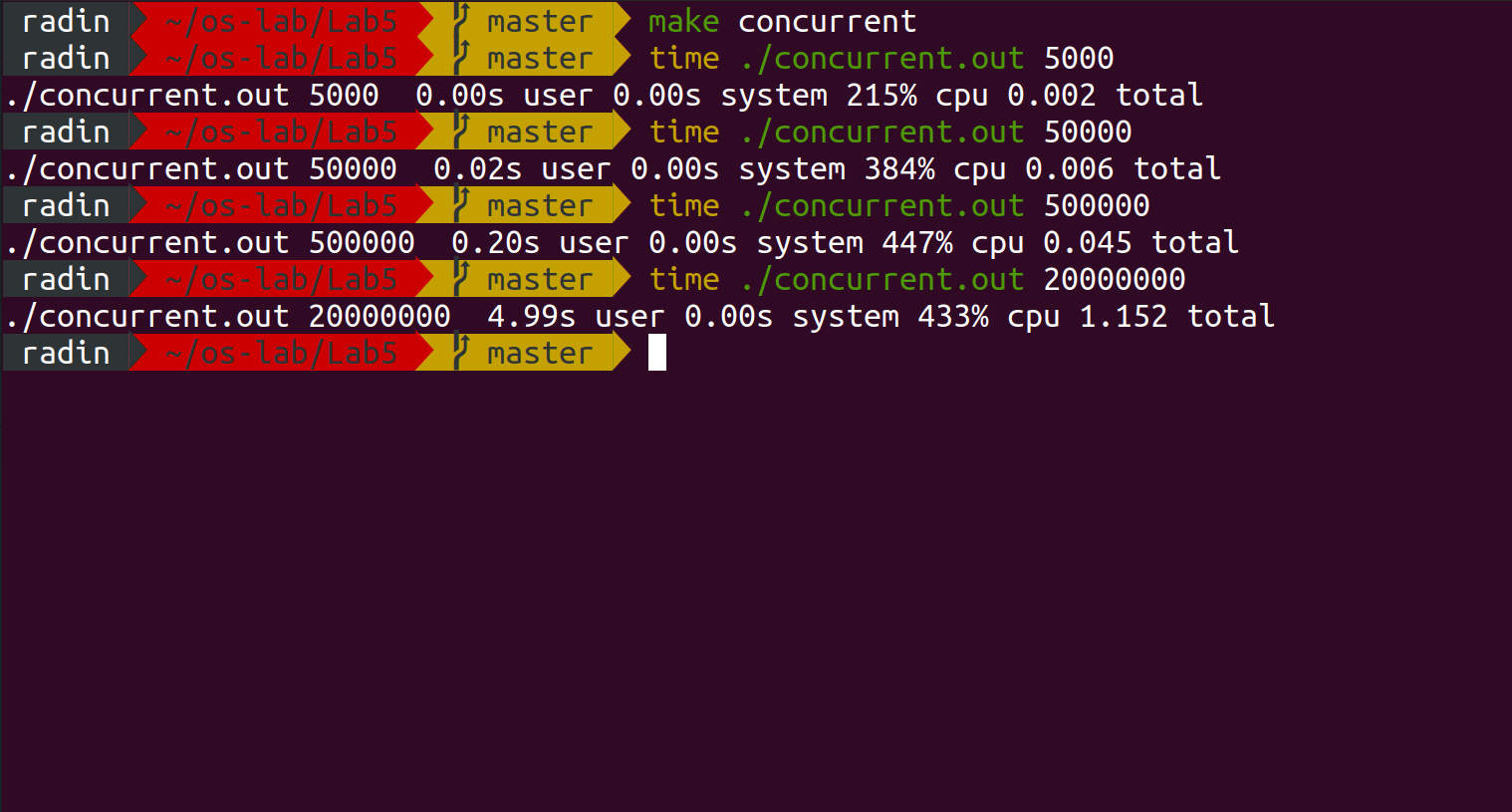
**شکل (۱) – نمودار هیستوگرام در ۲ حالت به ازای نمونه‌ی ۱۰۰ تایی**

1. زمان اجرای برنامه در حالت سریال به ازای تعداد نمونه‌های مختلف:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| تعداد نمونه | ۵۰۰۰ | ۵۰۰۰۰ | ۵۰۰۰۰۰ | ۲۰۰۰۰۰۰۰ |
| زمان اجرا | 7ms | 20ms | 120ms | 4150ms |

1. زمان اجرای برنامه در حالت هم‌روند به ازای تعداد نمونه‌های مختلف:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| تعداد نمونه | ۵۰۰۰ | ۵۰۰۰۰ | ۵۰۰۰۰۰ | ۲۰۰۰۰۰۰۰ |
| زمان اجرا | 2ms | 6ms | 45ms | 1152ms |



**شکل (۲) – زمان‌های اجرا در حالت هم‌روند**

**شکل (۲) – زمان‌های اجرا در حالت سریال**

1. بله. متغیر hist[counter + 12] می‌تواند ایجاد race condition کند. برای جلوگیری از این اتفاق می‌توان با استفاده از spin lock ها و یا سمافور، انحصار متقابل در حین دسترسی به آرایه‌ی hist ایجاد کرد. به عنوان مثال می‌توان به ازای هر خانه‌ی hist یک سمافور با مقدار اولیه یک ایجاد کرد و در هنگام ایجاد تغییر در آن، به شکل زیر کد را بازنویسی کرد:

sem\_wait(&sem[counter + 12]);

hist[counter + 12]++;

sem\_post(&sem[counter + 12]);

1. میزان افزایش سرعت در حالت هم‌روند:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| تعداد نمونه | ۵۰۰۰ | ۵۰۰۰۰ | ۵۰۰۰۰۰ | ۲۰۰۰۰۰۰۰ |
| افزایش سرعت | 5ms = 71% | 14ms = 70% | 75ms = 62% | 2998ms = 72% |