نقش پروگرام چند رشته‌یی در برنامه‌نویسی

پوهنیار محمد سلیم همدرد

دیپارتمنت انجینری نرم‌افزار، پوهنځی کمپیوتر ساینس، پوهنتون کابل، کابل، افغانستان

ایمیل: salimhamdard12345@gmail.com

چکیده

در این مقاله تحقیقی، نقش اساسی، مشکلات و موضوعات مربوط به یک پروگرام چند رشته‌ای مورد بحث و مطالعه قرار گرفته است. لسان‌های مختلف پروگرام نویسی به شمول پایتون به چند رشته تقسیم کردن یک پروگرام را تقویت می‌کند. پروگرام چند رشته‌ای دارایی دو یا چندین رشته که می‌تواند هم‌زمان اجرا شود، می‌باشد. هر رشته می‌تواند کار جداگانه در عین وقت انجام دهد که باعث استفاده بهتر منابع کمپیوتر می‌شود. نتایج این تحقیق که از روش تجربی و عملی استفاده شده است نشان می‌دهد که چند رشته کردن یک پروگرام به معنی انجام دادن چندین کار به‌صورت موازی در برنامه‌ها است که توسط آن کارهایی مشخص به هر رشته یک برنامه برای اجرا داده می‌شود. در پروگرام چند رشته‌ای، سیستم عامل برای اجرای هر رشته زمان پروسس بین چندین رشته یک برنامه تقسیم می‌کند.

اصطلاحات کلیدی: پروگرام؛ چند رشته کردن یک پروگرام؛ حالت رقابتی؛ هم‌آهنگ‌سازی رشته‌ها

Role of Multithreading in Program Development

Jr. Teaching Asstt. Mohammad Salim Hamdard

Department of Software Engineering, Faculty of Computer Science, Kabul University, Kabul, Afghanistan

Email: salimhamdard12345@gmail.com

Abstract

The purpose of this research article is to explore and analyze the main challenges and benefits of a multidisciplinary program that involves different programming languages, such as Python. One of the key features of these languages is the ability to create multi-threaded programs, which are programs that consist of two or more threads that can run concurrently. This allows each thread to perform a specific task independently and efficiently, making optimal use of the computer resources. The research methodology used in this article was based on experimental and practical approaches, and the findings indicate that multi-threading a program enables parallel execution of multiple tasks in the programs, by assigning each thread a certain task to complete. In a multi-threaded program, the operating system allocates the processing time among several threads of a program to execute each thread.

Keywords: Program; Multithreading; Race Condition; Threads Synchronization

مقدمه

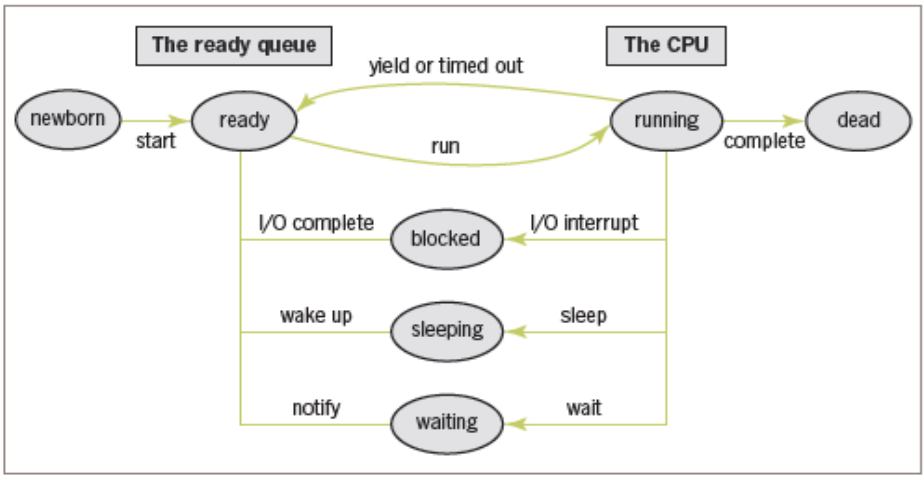
طوری‌که می‌دانیم الگوریتم یک پروسه محاسباتی را شرح می‌دهد و هم‌چنان الگوریتم یا پروگرام ضرورت به منابع دارد تا اجرا گردد. رشته (Thread) به مجموعه از دستورالعمل‌هایی در داخل یک پروگرام گفته می‌شود که به پروسسور (CPU) برای اجرا ارسال می‌گردد. پروگرام در حالت معمول دارای یک رشته می‌باشد؛ اما در بعضی حالات ضرورت احساس می‌شود تا پروگرام چندین کار در عین وقت انجام دهد. این نوع پروگرام باید چندین رشته داشته باشد که هم‌زمان باید اجرا گردد. اگر ما یک بازی کمپیوتری را در نظر بگیریم این نوع پروگرام ضرورت به چندین رشته دارد که باید هم‌زمان کار کنند. به‌طوری مثال یک رشته برای کارهایی گرافیکی رشته دوم برای تماس با استفاده‌کننده وغیره تا بتواند به‌شکل بهتر کار کنند.

لسان‌هایی مختلف برنامه‌نویسی مانند جاوا و پایتون تقسیم کردن یک پروگرام را به چندین رشته را حمایت می‌کند، دراین مقاله ما نقش تقسیم کردن پروگرام به چندین رشته در پایتون که یک زبان برنامه‌نویسی عصر امروزی است مورد بررسی و مطالعه قرار می‌دهیم.

این‌که پروگرام، پروسس، رشته، چند رشته کردن مفهوم و معنی مختلف در برنامه‌نویسی دارد لازم است تا تفاوت آن را واضح نماییم. پروگرام (Program) مجموعه از دستورالعمل‌های است که در یکی از زبان‌های برنامه‌نویسی نوشته و توسط کمپیوتر بعد از طی مراحل اجرا می‌گردد و یک کار را انجام می‌دهد (۱). پروگرام که در حالت اجرا باشد به نام پروسس (Process) یاد می‌شود. پروسس می‌تواند دارای یک یا چند رشته (Thread) باشد که هم‌زمان اجرا می‌گردد (۲،۱). چند رشته کردن (Multithreading) روشی است که به یک پروگرام یا پروسس اجازه می‌دهد که چندین کار توسط یک پروسسور (CPU) به‌شکل موازی یا هم‌زمان اجرا نماید.

در مجموع دو نوع رشته (Thread) وجود دارد: نوع اول آن مربوط سیستم عامل است که به‌نام (Kernel-level-thread) یاد می‌شود و نوع دوم آن توسط پروگرامر یا نویسنده برنامه در پروگرام ایجاد می‌شود که به نام (User-level-thread) یاد می‌گردد (۲).

هر رشته از یک تعداد حالت‌ها عبور می‌کند که با ایجاد رشته شروع و وقتی که رشته کار خود به‌شکل نورمال تکمیل کند و یا این‌که بنا بر بعضی دلایل دیگر از بین برده شود ختم می‌گردد (۳). شکل اول همه حالت‌های که رشته می‌تواند در یکی از این حالت‌ها قرار داشته باشد را نشان می‌دهد.



شکل ۱: حالت‌ها در زمان اجرای یک رشته (۱)

زمانی‌که رشته ایجاد می‌شود، رشته غیرفعال باقی می‌ماند و این حالت به‌نام حالت نوزاد (Newborn) یاد می‌گردد. طوری‌که در شکل مشاهده می‌گردد بعد از این‌که رشته ایجاد گردید، رشته منتظر دریافت پروسسور می‌باشد، تا اجرا گردد که این حالت به نام حالت آماده (Ready) یاد می‌شود. زمانی‌که رشته توسط پروسسور اجرا می‌گردد، این حالت به نام حالت اجرا(Running) یاد می‌گردد. بعد از این‌که رشته پروسسور را دریافت کرد، امکان دارد به‌شکل نورمال تا آخر اجرا گردد و یا این‌که بنا بر دلایل مختلف پروسسور از آن گرفته شود. به‌طوری مثال اجرای رشته برای چند ثانیه قصداً ایستاده می‌شود و یا این‌که رشته به دلیل انتظار برای وقوع یک حادثه اجرا نمی‌شود. حالت آخری که رشته کار خود را موفقانه تکمیل کند و یا قصداً ختم گردد، این حالت به‌نام حالت مرده (Dead) یاد می‌گردد (۳،۱).

اهداف تحقیق

در این تحقیق، نقش چند رشته کردن یک پروگرام در برنامه‌نویسی را مطالعه می‌نماییم. برای رسیدن به این هدف، اهداف ذیل را در نظر می‌گیریم:

1. توضیح فواید و چالش‌های یک پروگرام که دارای چندین رشته باشد و اهمیت آن در برنامه نویسی؛
2. توضیح میکانیزم‌های رفع چالش ناشی از برنامه‌نویسی چند رشته‌ای؛

سؤالات تحقیق

۱. فواید و چالش‌های یک پروگرام که دارای چندین رشته باشد کدام‌اند؟

۲. چگونه می‌توانیم، رشته‌ها در داخل یک پروگرام هم‌آهنگ کنیم تا به نوبت از قدرت محاسباتی پروسسور استفاده نماید؟

روش تحقیق

در این مقاله تحقیقی، از روش تجربی و عملی استفاده گردیده است. با استفاده از این روش ابتدا چالش‌های که در پروگرام‌نویسی چند رشته‌ای اتفاق می‌افتد مورد مطالعه قرار گرفته و بعد از مشخص کردن چالش‌ها، به‌مطالعه راه‌حل این چالش‌ها پرداخته شده است. با استفاده از یک سناریویی مشخص، چالش پیش‌بینی شده در پروگرام چند رشته‌ای به‌گونه عملی کدنویسی شده و از این طریق نشان داده شده که چالش رخ می‌دهد. در مرحله بعد به رفع چالش به‌صورت عملی پرداخته شده است.

فرضیه (Hypothesis) این تحقیق عبارت از هم‌آهنگی بین رشته‌ها در داخل یک پروگرام ما را قادر به تطبیق موفقانه برنامه‌نویسی چند رشته‌ای می‌کند.

طوری‌که در فرضیه تحقیق واضح گردید، هم‌آهنگ‌سازی بین رشته‌ها در داخل یک پروگرام می‌توان به حیث متحول مستقل و نتیجه (Output) پروگرام به حیث متحول وابسته تعریف نمود.

پیشینه تحقیق

یک پروسه یا پروگرام متشکل از یک و یا چندین رشته می‌باشد که به‌شکل موازی و هم‌زمان اجرا می‌گردد. رشته‌های مختلف در داخل یک پروسه به آسانی می‌تواند با هم منابع مانند میموری (Memory) شریک سازد؛ اما پروسه‌های جداگانه نمی‌تواند به‌طوری آسان با هم معلومات شریک نماید (٦).

در سیستم که دارای یک پروسسور باشد، پروسسور بین رشته‌های مختلف یک پروسه یا پروگرام تعویض می‌گردد. در سیستم که دارای پروسسور چند هسته‌یی (Multi-core Processor) باشد، رشته‌های مختلف یک پروگرام به‌طوری نسبتاً مستقل و بدون تعویض در هسته‌های مختلف اجرا می‌گردد (۳،٦).

برای ایجاد پروسه‌های جداگانه، در مقایسه با ایجاد کردن رشته‌های مختلف در داخل یک پروسه، به منابع بیش‌تر ضرورت می‌باشد. به‌طوری کلیThread برای کارهای کوچک و پروسه‌ها برای کارهای سنگین‌تر استفاده می‌شوند (۷). خوبی Thread در این است که پروگرام‌نویسی آن‌ها آسان‌تر است، در حالی‌که پروسه‌ها مشکل بیش‌تری را ایجاد می‌کند، به‌خاطر این‌که معلومات در بین پروسه‌ها به راحتی به اشتراک گذاشته نمی‌شود.

پروگرام چند رشته‌ای، در مقایسه به ایجاد پروسه‌هایی جداگانه دارایی فواید زیاد می‌باشد. مدیریت رشته‌ها نظر به پروسه‌ها آسان‌تر است، رشته‌ها در داخل یک پروسه می‌تواند به آسانی در تماس شوند و معلومات شریک کند. پروگرام چند رشته‌ای در هر حالت می‌تواند پاسخ‌گو باشد، چون هر رشته آن کار جداگانه و هم‌زمان اجرا می‌کند (۴). از همه مهم‌تر این روش باعث سرعت در کارایی (performance) می‌شود.

طوری‌که تذکر یافت، رشته‌های مختلف پروگرام با هم معلومات شریک می‌کند. بنابراین تغییر هم‌زمان دیتا توسط چندین رشته یک چالش اساسی است که باعث می‌گردد تا نتیجه (Output) دقیق به دست

نمی‌آید. در بعضی حالات یک رشته پروگرام به‌شکل دوام‌دار از منابع استفاده می‌کند و باعث می‌گردد تا رشته دیگر پروگرام به منابع دست‌رسی نداشته باشد که این حالت به نام (Deadlock) یاد می‌گردد (۵). برای از بین بردن این چالش‌ها از روش‌های استفاده می‌کنیم تا در یک وقت؛ یک رشته پروگرام؛ توان دست‌رسی به منابع را داشته باشد.

ایجاد کردن رشته در پایاتون (Thread Creation in Python)

برای این‌که رشته‌ها را در پایتون ایجاد کنیم از مادیول به نام threading استفاده می‌نماییم. این مادیول دارای همه منابع که برای ایجاد و مدیریت کردن پروگرام چند رشته‌ای ضروری است، می‌باشد. طریقه آسان برای ایجاد رشته این است که اول مادیول‌های که ما از آن استفاده می‌کنیم، داشته باشیم و بعد از آن با استفاده از کلاس threading.Thread رشته بسازیم (۹). یک مثال عملی که چگونه می‌توانیم رشته ایجاد کنیم در نظر گرفته شده است.

```
import threading
import time
def print_name(threadName, delay):
  count=0
  while count<3:
    time.sleep(delay)
    count +=1
    print ("%s: %s" %(threadName,time.ctime(time.time())))
t1=threading.Thread(target=print_name,args=("Thread-1",1))
t2=threading.Thread(target=print_name,args=("Thread-2",3))
t1.start()
t2.start()
t1.join()
t2.join()
print("Done")
```

طوری‌که در پروگرام مشاهده می‌گردد، با استفاده از کلاس threading.Thread دو رشته ایجاد نمودیم t1 و t2 ، بعد از ایجاد، ما باید میتود را مشخص نماییم که توسط رشته‌ها اجرا گردد، در این پروگرام هر دو رشته یک میتود print_name را اجرا می‌کند. میتود(start) که مربوط کلاس threading.Thread استفاده می‌کنیم تا رشته فعال گردد و میتود (join) باعث می‌شود تا اول هر دو رشته مکمل اجرا گردد و بعد از اجرای آن رشته اساسی (Main Thread) به فعالیت آغاز کند (۹). نظر به پارامترهای که ما به میتود print_name انتقال می‌دهیم نتیجه این پروگرام بعد از اجرا قرار ذیل است:

```
Thread-1: Thu Feb 16 14:13:42 2023
Thread-1: Thu Feb 16 14:13:43 2023
Thread-2: Thu Feb 16 14:13:44 2023
Thread-1: Thu Feb 16 14:13:44 2023
Thread-2: Thu Feb 16 14:13:47 2023
Thread-2: Thu Feb 16 14:13:50 2023
Done
```

فرض کنید که ما ضرورت داریم تا تعداد زیاد از Thread ایجاد کنیم، در این‌صورت روش قبلی یک روش مناسب نیست، چون ساختن و مدیریت Thread مطابق به روش قبلی یک کار نسبتاً دشوار است. در چنین حالات ما می‌توانیم از Thread Pool استفاده نماییم، که گروپ از Thread های از قبل ایجاد شده است که در وقت ضرورت می‌تواند استفاده شود. این یک روش مناسب و ساده‌تر می‌باشد؛ زیرا در یک سطر کود می‌توانیم چندین رشته ایجاد کنیم (۱۰). برای عملی ساختن این روش در پایتون از مادیول Thread Pool Executor استفاده می‌نماییم، در ابتدا تعداد Thread های که می‌خواهیم ایجاد کنیم مشخص می‌نماییم، بعد از آن توسط میتود (submit) به هر Thread وظیفه اختصاص می‌دهیم تا آن را اجرا نماید. در مثال‌هایی بعدی از این روش استفاده شده است.

حالت رقابتی (Race Condition)

طوری‌که قبلاً تعریف نمودیم، Multithreading روشی است که توسط آن می‌توان چندین Thread را در CPU (پروسسور) هم‌زمان اجرا کرد (۳). اجرای هم‌زمان این Thread ها به این معنی است که به نوبت از قدرت محاسباتی پروسسور استفاده می‌کند، بدون از این‌که تأثیر با هم دیگر داشته باشد.

حالتی که چندین رشته یک پروگرام تلاش کند تا هم‌زمان به معلومات دست‌رسی داشته باشد و در آن تغییرات ایجاد کند، درحالی‌که همه رشته‌ها نتواند تغییرات ایجاد شده را ذخیره کند به‌نام Race Condition یاد می‌شود (۱۱). این حالت باعث نتیجه غیر دقیق می‌شود که سبب هزینه بسیاری می‌گردد. برای مثال فرض نمائید دو رشته (Thread) هم‌زمان کوشش می‌کنند تا مقدار مشخص صفر را خوانده و به اندازه یک واحد به آن اضافه کنند. در نتیجه انتظار دارم مقدار حاصل شده ۲ باشد؛ ولی نتیجه پروگرام عدد ۱ است.

```
import time
import threading
from concurrent.futures import ThreadPoolExecutor
class Counter:
  def __init__(self):
     self.value=0
   def update(self,name):
     print(f"Update Started by {name}")
     val=self.value
     val +=1
     time.sleep(1)
     self.value=val
     print(f"Update Ended by {name}")
counter=Counter()
with ThreadPoolExecutor(max_workers=3) as executor:
   executor.submit(counter.update,"Thread-0")
   executor.submit(counter.update,"Thread-1")
print(counter.value)
```

نتیجه این پروگرام قرار ذیل است:

```
Update Started by Thread-0
Update Started by Thread-1
Update Ended by Thread-0
Update Ended by Thread-1
1
```

هم‌آهنگ‌سازی رشته‌ها (Threads Synchronization)

طوری‌که واضیح نمودیم، پروگرام چند رشته‌ای وقتی نتیجه دقیق دارد که Thread ها را هم‌آهنگ نماییم تا به نوبت از قدرت محاسباتی پروسسور استفاده نماید. برای این کار ما در یک وقت معین به یک Thread دست‌رسی به منابع می‌دهیم به این معنی که در یک وقت یک Thread بتواند در معلومات تغییر ایجاد کند، این روش به‌نام هم‌آهنگی رشته‌ها (Threads Synchronization) یاد می‌شود (۸).

مادیول threading هم‌آهنگ کردن Thread ها را حمایت می‌کند، برای انجام این کار در ابتدا ما ضرورت داریم تا یک Object کلاس Lock بسازیم، بعد از آن با استفاده این Object توسط میتود acquire دست‌رسی به منابع و معلومات حاصل نماییم. هم‌چنان زمانی‌که یک Thread کار خود را تکمیل کند، توسط میتود release منابع را از دست می‌دهد تا یک Thread دیگر به منابع دست‌رسی حاصل نماید (۹). نظر به مثال قبلی ما Thread ها را چنین هم‌آهنگ می‌کنیم:

```
import time
import threading
from concurrent.futures import ThreadPoolExecutor
class Counter:
  def __init__(self):
      self.value=0
      self.lock=threading.Lock()
   def update(self,name):
      print(f"Update Started by {name}")
      self.lock.acquire()
      val=self.value
      val +=1
      time.sleep(1)
      self.value=val
      self.lock.release()
      print(f"Update Ended by {name}")
counter=Counter()
with ThreadPoolExecutor(max_workers=3) as executor:
   executor.submit(counter.update,"Thread-0")
   executor.submit(counter.update,"Thread-1")
print(counter.value)
```

نتیجه این پروگرام قرار ذیل است و این یک نتیجه درست می‌باشد:

```
Update Started by Thread-0
Update Started by Thread-1
Update Ended by Thread-0
Update Ended by Thread-1
2
```

مناقشه و نتیجه‌گیری

طوری‌که بحث شد یک پروگرام چند رشته‌ای که هر رشته آن می‌تواند یک کار جداگانه اجرا نماید، دارایی اهمیت خاص در برنامه‌نویسی می‌باشد. مدیریت کردن رشته‌ها در داخل یک پروسه آسان‌تر است و می‌تواند به آسانی در تماس شوند و معلومات شریک نماید، نظر به این‌که پروگرام‌ها یا پروسس‌هایی جداگانه ساخته شود. پروگرام چند رشته‌ای در هر حالت می‌تواند جواب‌گو باشد و کارایی خوب دارد. هم‌چنان برنامه‌هایی زیاد ضرورت دارد، تا هر رشته آن به‌شکل موازی و هم‌زمان اجرا شوند تا در هر حالت جواب‌گو باشد.

نتایج که از این تحقیق به‌دست آمده نشان می‌دهد که یک سلسله چالش‌ها مانند Race Condition (حالتی که چندین رشته یک پروگرام تلاش کند تا هم‌زمان به معلومات دست‌رسی داشته باشد و در

حالی‌که همه رشته‌ها نتواند تغییرات ایجاد شده را ذخیره کند) و یا حالت Deadlock (حالتی که یک رشته پروگرام به‌شکل دوام‌دار از منابع استفاده می‌کند و اجازه نمی‌دهد تا رشته دیگر پروگرام به منابع دست‌رسی داشته باشد) وجود دارد. چالش‌های ذکر شده بالای نتیجه پروگرام اثرات منفی دارد، پروگرام یک نتیجه یا Output درست و دقیق نمی‌دهد و یا این‌که اجرای پروگرام قبل از این‌که موفقانه تکمیل گردد، متوقف می‌شود.

برای این‌که از Multithreading به درستی استفاده نماییم و همه مزایایی آن در برنامه‌نویسی تطبیق نماییم، باید از وقوع این چالش‌ها جلوگیری نماییم. برای ایجاد رشته‌ها در لسان برنامه‌نویسی پایتون، مادیول threading استفاده می‌نماییم. برای جلوگیری از وقوع چالش‌ها مانند Race Condition و یا حالت Deadlock ضرورت به یک میکانیزم است تا بتوانیم Thread ها را هم‌آهنگ نماییم تا به نوبت از قدرت محاسباتی پروسسور استفاده نماید، برای این کار ما در یک وقت معین به یک Thread دست‌رسی به منابع می‌دهیم به این معنی که در یک وقت یک Thread بتواند در معلومات تغییر ایجاد کند.

منابع

1. Lambert KA. Fundamentals of Python: First Programs Second Edition. 2017. 352–390 p.
2. Carver RH, Tai KC. Modern Multithreading: Implementing, Testing, and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs. 2005. 1–465 p.
3. Goel N, Laxmi V, Saxena A. Handling Multithreading Approach Using Java. Int J Comput Sci Trends Technol [Internet]. 2015;3(2):24–31. Available from: http://www.ijcstjournal.org/volume-3/issue-2/IJCST-V3I2P5.pdf
4. Rugina R, Rinard M. Pointer analysis for multithreaded programs. SIGPLAN Not (ACM Spec Interes Gr Program Lang. 1999;34(5):77–90.
5. Palach J. Parallel Programming with Python [Internet]. Packt Publishing. 2014. 124 p. Available from: www.it-ebooks.info
6. Kavi K. Multithreading Implementations The University of Texas at Arlington. 2013;(September 1998).
7. Eggen R. Python: Thread or Process. 2019 Hawaii Univ Int Conf [Internet]. 2019; Available from: https://huichawaii.org/wp-content/uploads/2019/07/Eggen-Roger-2019-STEM-HUIC.pdf
8. Farah T, Shelim R, Zaman M, Hassan MM, Alam D. Study of race condition: A privilege escalation vulnerability. WMSCI 2017 - 21st World Multi-Conference Syst Cybern Informatics, Proc. 2017;2(June):100–5.
9. Zaccone G. Python parallel programming cookbook: master efficient parallel programming to build powerful applications using Python [Internet]. 2015. 286 (est.). Available from: http://proquest.safaribooksonline.com/?fpi=9781785289583
10. Brownlee J. Python ThreadPool Jump-Start [Internet]. Kindle E, editor. 2022. 76 p. Available from: https://www.goodreads.com/book/show/61922939-python-threadpool-jump-start
11. Carr S, Mayo J, Shene C-K. Race Conditions: A Case Study. J Comput Sci Coll [Internet]. 2001;17(1):90–105. Available from: http://dl.acm.org/citation.cfm?id=772488.772504