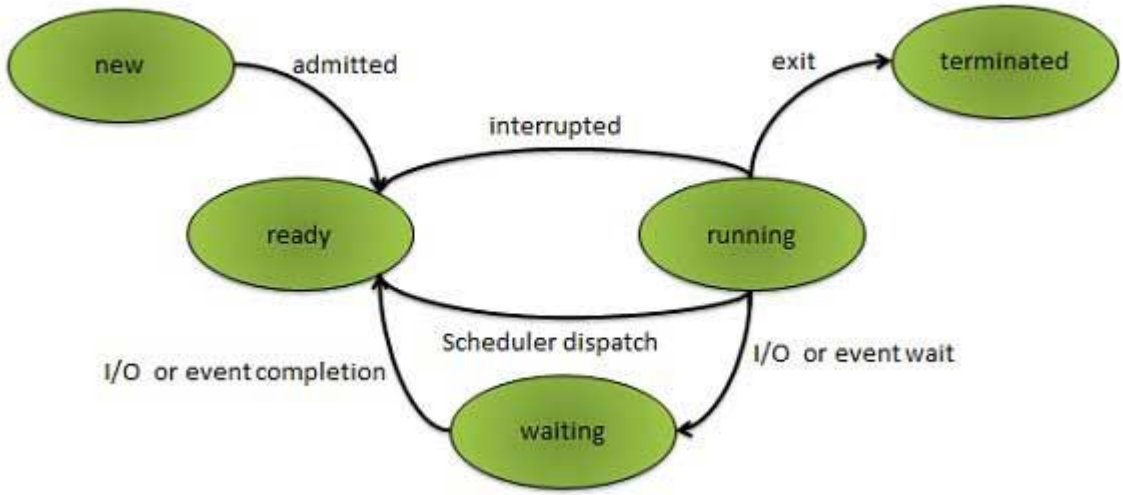
Hoja de trabajo No. 5

Realizar: Programa de uso de colas.
Realizarse: grupos de tres personas

Objetivos:
a. Simulación DES (Discrete Event Simulation) usando el módulo SimPy.
b. Utilización de colas con la clase Resources y Container de SimPy.

Simulación:
Simulación de corrida de programas en un sistema operativo de tiempo compartido (el procesador se comparte por una porción de tiempo entre cada programa que se desea correr). En la terminología de sistemas operativos se llama "proceso" a un programa que se ejecuta. La siguiente figura muestra el ciclo de vida de un proceso dentro del sistema operativo:



new: el proceso llega al sistema operativo pero debe esperar que se le asigne memoria RAM. En nuestra simulación el proceso solicitará una cantidad de memoria (número entero al azar entre 1 y 10). Si hay memoria disponible puede pasar al estado de ready. En caso contrario permanece haciendo cola, esperando por memoria. NOTA: la cantidad de memoria disponible se disminuye con la cantidad de memoria que empleará el proceso que logró obtenerla.

ready: el proceso está listo para correr pero debe esperar que lo atienda el CPU. El proceso tiene un contador con la cantidad de instrucciones totales a realizar (número entero al azar entre 1 y 10). Cuando se desocupa el CPU puede pasar a utilizarlo.

running: el CPU atiende al proceso por un tiempo limitado, suficiente para realizar solamente 3 instrucciones (esto será configurado, ya que podrá crecer o reducirse). Al completarse el tiempo de atención el proceso es retirado del CPU. Se debe actualizar el contador de instrucciones a realizar, disminuyendo las 3 instrucciones que ejecutó en esta oportunidad. Nota: para simplificar la simulación, si la cantidad de instrucciones a realizar es menor que 3, el procesador de igual manera dedica un ciclo completo para realizarlas.

Al finalizar la atención del CPU puede ocurrir:
a)Terminated: Si el proceso ya no tiene instrucciones por realizar entonces pasa al estado "terminated" y sale del sistema.
b)Waiting: al dejar el CPU se genera un número entero al azar entre 1 y 2[1]. Si es 1 entonces pasa a la cola de Waiting para hacer operaciones de I/O (entrada/salida). Al dejar esa cola regresa a "ready".
c)Ready: al dejar el CPU y el número generado al azar es 2 entonces se dirige nuevamente a la cola de "ready".

[1] Esta es una simplificación de la simulación. En la realidad se genera una interrupción para las instrucciones de I/O

Parámetros de la simulación:

La creación de procesos sigue una distribución exponencial con intervalo = 10. Recordar que entonces se generan los números al azar para simular la llegada de procesos con: random.expovariate(1.0 / interval)

La cantidad de memoria RAM de la computadora es 100. Se usa un recurso tipo Container[2] para simular la memoria. Se debe hacer cola para solicitar la cantidad de memoria necesaria y se permanece en la cola hasta que se obtenga esa cantidad. Al finalizar un proceso se regresa la cantidad de memoria que utilizó[3]. Ejemplo de definición de un Container:

```
RAM = simpy.Container(env, init=100, capacity=100)
y para tomar memoria del RAM: RAM.get(memoria)
y para devolver memoria al RAM: RAM.put(memoria)
en donde memoria es la cantidad de memoria que se está requiriendo del RAM.
```

La velocidad del CPU se modela con que atiende un proceso en una 1 unidad de tiempo, lo cual permite realizar tres instrucciones. Esto debe ser variable, ya que podríamos decir luego que tenemos un procesador más rápido que ejecuta más instrucciones en esa unidad de tiempo. El CPU es modelado con una cola tipo Resource (capacidad = 1, es decir un solo CPU).

NOTA: para la generación de los números al azar utilice una semilla para que se genere siempre la misma secuencia y así se puedan hacer comparaciones cuando se cambien los parámetros de la simulación: random.seed(RANDOM_SEED)

Puede utilizar como base los ejemplos colocados en Canvas en los contenidos del curso.

Tareas:

a. Hacer el programa de simulación y usarlo con 25 procesos, luego con 50 procesos, con 100, 150 y 200 procesos. Su programa debe mostrar el promedio de tiempo que está el proceso en la computadora en cada caso y la desviación standard. Haga gráfica con número de procesos y tiempo promedio.
b. Vuelva a correr su simulación, pero ahora los procesos llegar más rápido, es decir en intervalos de 5. Calcule los tiempo promedio para las mismas cantidades de procesos: 25,50,100,150 y 200. Repita lo mismo para intervalos de 1 (mucha carga de trabajo). Haga gráfica con número de procesos y tiempo promedio.
c. Revise las gráficas y trate de reducir el tiempo promedio. Pruebe con:
    i. incrementar la memoria a 200,
    ii. luego con poner la memoria nuevamente a 100, pero tener un procesador más rápido (es decir que ejecuta 6 instrucciones por unidad de tiempo),
    iii. luego regrese a la velocidad normal procesador pero emplee 2 procesadores.

    Haga gráficas para cada cambio con las cantidades de 25,50,100,150 y 200 procesos (e intervalos de 10, 5, 1). Decida cuál es la mejor estrategia para reducir el tiempo promedio de ejecución de los procesos, justifique su respuesta.
d. Realice la misma simulación pero con 2 procesadores, ¿Qué nota de diferente?, aumente también la cantidad de memoria disponible y verifique si el cambio tiene algún efecto en los resultados.

Debe subir a la plataforma el programa realizado, las gráficas realizadas y su decisión sobre cual estrategia seguir para reducir el tiempo promedio de corrida de los procesos.

[2] http://simpy.readthedocs.org/en/latest/topical_guides/resources.html#res-type-container

[3] Esta es otra simplificación de la simulación, ya que el proceso está deteniendo la cola hasta que pueda ser atendido.

Calificación: su programa debe funcionar para ser calificado.

Aspecto	Puntos
Programa de simulación funcionando.	50
Graficas realizadas (deben estar bien identificadas)	35
Estrategia recomendada para reducir el tiempo promedio de los procesos.	10
Control de versiones en git o sistema similar.	5
TOTAL:	100