Universidad Tecnológica Nacional – FR San Rafael
Sistemas Operativos – TP 6
Respuestas teóricas

1) ¿En cuál de los tres sistemas se necesita que los procesos estén relacionados?
- En multiprocessing.Queue (Python): sí, deben ser procesos relacionados que comparten el objeto Queue crea
- En POSIX MQ y System V MQ: no es necesario que estén relacionados; se identifican globalmente (nombre

2) ¿En qué casos las colas están identificadas por un nombre y en qué casos por un ID?
- POSIX Message Queue: por un nombre estilo "/nombre" (ej., /tp6_posix_queue).
- System V Message Queue: por un identificador (ID) obtenido con msgget() a partir de una key.
- multiprocessing.Queue: no tiene nombre/ID del sistema; es un objeto en memoria compartido entre procesos

3) ¿Cuál de los tres sistemas ofrece prioridad en los mensajes?
- POSIX MQ: sí, soporta prioridad explícita (parámetro "prio" en mq_send / devuelve prioridad en mq_receiv
- System V MQ: no tiene prioridad de mensajes; se puede seleccionar por mtype, pero el orden por defecto es
- multiprocessing.Queue: FIFO puro, sin prioridad.

4) ¿Qué pasa si el consumidor es más lento que el productor?
- POSIX MQ: la cola se llena (según mq_maxmsg). Si se llena, mq_send bloquea salvo que esté O_NONBLO
- System V MQ: la cola se llena (msg_qnum / límites del sistema). msgsnd bloquea salvo IPC_NOWAIT (ento
- multiprocessing.Queue: si se configuró maxsize y se llena, put() bloquea (o lanza Full si block=False/timeou

5) ¿Qué ocurre si la cola está llena y el productor intenta enviar un mensaje?
- POSIX MQ: mq_send bloquea hasta que haya espacio; con O_NONBLOCK → error (EAGAIN).
- System V MQ: msgsnd bloquea; con IPC_NOWAIT → error (EAGAIN).
- multiprocessing.Queue: put() bloquea; con block=False o timeout → levanta queue.Full.