Programación concurrente

Práctica repaso - Memoria compartida

Semáforos

1)

a)

```
1) a)

# Versión 1
sem mutex = 1, turno_de[P] = ([P] 0);
Cola cola;

Process Persona [id = 0..P-1] {
      P(mutex);
      if(!cola.empty()) {
            cola.push(id);
            V(mutex);
            P(turno_de[id]);
      } else {
            cola.push(id);
            V(mutex);
      }

      UsarTerminal();

      P(mutex);
      cola.pop();
      if(!cola.empty()) {
            int prox_id;
            cola.peek(prox_id);
            V(turno_de[prox_id]);
      }
```

```
      V(mutex);
}

# Versión 2
sem mutex = 1, turno_de[P] = ([P] 0);
Cola cola;
bool libre = true;

Process Persona [id = 0..P-1] {
      P(mutex);
      if(!libre) {
            cola.push(id);
            V(mutex);
            P(turno_de[id]);
      } else {
            libre = false;
            V(mutex);
      }

      UsarTerminal();

      P(mutex);
      if(!cola.empty()) {
            int prox_id;
            cola.pop(prox_id);
            V(turno_de[prox_id]);
      } else {
            libre = true;
      }
      V(mutex);
}
```

b)

```
1) b)

sem mutex = 1, turno_de[P] = ([P] 0);
Cola cola, cola_terminales[t] = terminales;

Process Persona [id = 1..P] {
      P(mutex);
      if(terminales.empty()) {
            cola.push(id);
            V(mutex);
            P(turno_de[id]);
      }
      Terminal terminal_a_usar;
      cola_terminales.pop(terminal_a_usar);
      V(mutex);

      UsarTerminal(terminal_a_usar);

      P(mutex);
      cola_terminales.push(terminal_a_usar);
      if(!cola.empty()) {
            int prox_id;
            cola.pop(prox_id);
            V(turno_de[prox_id]);
      } else {
            V(mutex);
      }
}
```

2)

```
2)

sem mutex = 1, mutex_worker_finalizados = 1;
Transaccion transacciones[10000];
int transaccion_a_validar = 0, workers_finalizados = 0;
int cantidad_transacciones_resultado[0..9] = ([0..9] 0);

Process Worker [id = 0..6] {
     Transaccion t;
     int resultado;
     int cantidad_transacciones_local[0..9] = ([0..9] 0);

     P(mutex);
     while(transaccion_a_validar < 10000) {
          t = transacciones[transaccion_a_validar];
          transaccion_a_validar ++;
          V(mutex);

          resultado = Validar(t);
          cantidad_transacciones_local[resultado] ++;
          P(mutex);
     }
     V(mutex);
     for [i = 0..9] {
          cantidad_transacciones_resultado[i] +=
cantidad_transacciones_local[i];
     }
     P(mutex_worker_finalizados);
     workers_finalizados ++;
```

```
      cant
      if(workers_finalizados == 7) {
            informar(cantidad_transacciones_resultado);
      }
      V(mutex_worker_finalizados);
}
```

3)

```
3)

sem mutex = 1, prox_id[U] = ([U] 0), reponer = 0,
reposicion_completa = 0;
bool libre = true;
Cola cola;
int cant_latas = 100;

Process Usuario [id = 0..U-1] {
      P(mutex);
      if(!libre) {
            cola.push(id);
            V(mutex);
            P(turno_de[id]);
      } else {
            libre = false;
            V(mutex);
      }

      if(cant_latas == 0) {
            V(reponer);
```

```
            P(reposicion_completa);
      }
      cant_latas --;

      P(mutex);
      if(!cola.empty()) {
            int prox_id;
            cola.pop(prox_id);
            V(turno_de[prox_id]);
      } else {
            libre = true;
      }
      V(mutex);
}

Process Repositor {
      while(true) {
            P(reponer);
            cant_latas = 100;
            V(reposicion_completa);
      }
}
```

Monitores

1)

```
1)

Monitor AccesoAMaquina {
	cond hay_persona, turno_de[N], maquina_libre;
	Cola cola;
	bool libre = true;



	Procedure solicitar_acceso(id: in int, edad: in int;,
esta_embarazada: in bool) {
		cola.insertar_ordenado(id, edad, esta_embarazada);
		signal(hay_persona);
		wait(turno_de[id]);
	}

	Procedure dar_acceso() {
		if(cola.empty()) {
			wait(hay_persona);
		}

		if(!libre) {
			wait(maquina_libre);
		}
		libre = false;

		int prox_id;
		cola.pop(prox_id);
```

```
            signal(turno_de[prox_id]);
      }

      Procedure liberar() {
            libre = true;
            signal(maquina_libre);
      }
}

Process Persona [id = 0..N-1] {
      AccesoAMaquina.solicitar_acceso(id, getEdad(id),
estaEmbarazada(id));
      Votar();
      AccesoAMaquina.liberar();
}

Process AutoridadDeMesa {
      while(true) {
            AccesoAMaquina.dar_acceso();
      }
}
```

2)

```
2)

Monitor Equipo [0..4] {
     int cant_llegaron = 0, cant_terminaron = 0,
cant_ejemplares_equipo = 0;
     cond llegaron_todos, finalizaron_todos;

     Procedure llegar() {
          cant_llegaron ++;
          if(cant_llegaron == 4) {
               signalall(llegaron_todos)
          } else {
               wait(llegaron_todos);
          }
     }

     Procedure finalizar(cant_ejemplares_vendidos: in int,
cant_ejemplares_totales: out int) {
          cant_terminaron ++;
          cant_ejemplares_equipo += cant_ejemplares_vendidos;
          if(cant_terminaron == 4) {
               signalall(finalizaron_todos)
          } else {
               wait(finalizaron_todos);
          }
          cant_ejemplares_totales = cant_ejemplares_equipo;
     }
}
```

```
Process Vendedor [0..19] {
     int equipo, cant_ejemplares_vendidos,
cant_ejemplares_totales;
     getEquipo(equipo);

     Equipo[equipo].llegar();
     vender_ejemplares_producto(equipo, cant_ejemplares_vendidos);
     Equipo[equipo].finalizar(cant_ejemplares_vendidos,
cant_ejemplares_totales);
}
```

3)

```
3)

Monitor AccesoAlPaso {
     bool libre = true;
     int cant_esperando = 0;
     cond paso_libre;

     Procedure llegar() {
          if(!libre) {
               cant_esperando ++;
               wait(paso_libre);
               cant_esperando --;
          } else {
               libre = false;
          }
     }
```

```
      Procedure irse() {
            if(cant_esperando > 0) {
                  signal(paso_libre);
            } else {
                  libre = true;
            }
      }
}

Process Escalador [id = 0..29] {
      AccesoAlPaso.llegar();
      pasar_paso();
      AccesoAlPaso.irse();
}
```