Première partie

Tests unitaires

Module Réseau / réception – `Test 1` – Réception du message configuration, nominal

Description

Comportement lorsqu'un message de configuration est reçu.

Résultat attendu

Le système reçoit un message qui correspond au protocole :

```
CONFIG
a
b
c
d
e
```

Les valeurs dans le segment de mémoire partagée `settings` sont mises à jour. Un message doit être envoyé dans la boit aux lettre des évènements.

Module Réseau / réception – `Test 2` – Réception du message configuration, erreur

Description

Comportement lorsqu'un message de configuration erroné est reçu. Il peu s'agit d'un nombre de paramètre trop peu important, ou de paramètres avec des valeurs non valides.

Résultat attendu

Le message est ignoré. Un message doit être ajouté à la boite aux lettres de évènements, précisant l'erreur.

Module Réseau / réception – `Test 3` – Réception du message de lancement, nominal

Description

Comportement lorsqu'un message de lancement de l'application est reçu, après avoir reçu une configuration valide précédemment.

Résultat attendu

Le sémaphore `boxHandlingRequest`, contrôlant la synchronisation de la chaine de production doit être libéré. Un message doit être ajouté à la boite aux lettre des évènements.

Module Réseau / réception – `Test 4` – Réception du message de lancement, erreur

Description

Comportement lorsqu'un message de lancement de l'application est reçu, sans avoir reçu au préalable une configuration valide.

Résultat attendu

Le message est ignoré. Un message doit être ajouté à la boite aux lettres des évènements, précisant l'erreur.

Module Réseau / réception – `Test 5` – Réception du message d'arrêt de l'application, nominal

Description

Comportement lorsque le message d'arrêt de l'application est reçu, alors que l'application est en fonctionnement.

Résultat attendu

Une valeur doit être ajouté dans le segment de mémoire partagé `settings`, indiquant la fin de l'application.

Module Réseau / réception – `Test 6` – Réception du message d'arrêt de l'application, erreur

Description

Comportement lorsque le message d'arrêt de l'application est reçu, alors que l'application est en arrêt (par exemple avant le lancement, ou après une erreur et avant la reprise).

Résultat attendu

Le message est ignoré. Un message doit être ajouté à la boite aux lettres des évènements, précisant l'erreur.

Module Réseau / réception – `Test 7` – Réception du message de reprise, nominal

Description

Comportement lorsque le message d'arrêt de l'application est reçu, alors que l'application est en arrêt (après une erreur).

Résultat attendu

Un jeton est mis dans le sémaphore de synchronisation de la chaine de production (`boxHandlingRequest`). Un message est envoyé dans la boite aux lettres des évènements.

Module Réseau / réception – `Test 8` – Réception du message de reprise, erreur

Description

Comportement lorsque le message d'arrêt de l'application est reçu, alors que l'application est en fonctionnement normal, ou avant qu'une configuration valide ait été reçue.

Résultat attendu

Le message est ignoré. Un message doit être ajouté à la boite aux lettres des évènements, précisant l'erreur.

Module Réseau / réception – `Test 9` – Réception d'un message non défini dans le protocole

Description

Comportement lorsqu'un message non spécifié dans le protocole est reçu.

Résultat attendu

Le message est ignoré. Un message doit être ajouté à la boite aux lettres des évènements, précisant l'erreur.

Module Réseau / émission – `Test 10` – Envoi un message d'acceptation de pièce

Description

Comportement lorsqu'un message est reçu comme quoi un carton a été accepté par le système.

Résultat attendu

Un message est envoyé, précisant le nombre de pièces acceptées, et le nombre de pièces mises au rebut pour le remplissage de ce carton.

Module Réseau / émission – `Test 11` – Envoi d'un message d'alerte pour le nombre de pièces défectueuses

Description

Comportement lorsqu'un évènement est reçu comme quoi un trop grand nombre de pièce défectueuses a été traité par le système

Résultat attendu

Un message est envoyé, précisant que le système est stoppé en raison d'un trop grand nombre de pièces défectueuses.

Module Réseau / émission – `Test 12` – Envoi d'un message d'alerte quand la file d'attente est pleine.

Description

Comportement lorsqu'un évènements est reçu comme quoi la file d'attente d'impression est pleine.

Résultat attendu

Un message est envoyé, précisant que la file d'attente d'impression est vide.

Module Réseau / émission – `Test 13` – Envoi d'un message d'alerte lors d'une famine de pièces.

Description

Comportement lorsqu'un évènement est reçu comme quoi il y a une famine de pièce.

Résultat attendu

Un message est envoyé, précisant qu'une famine de pièce est survenu.

Module Réseau / émission – `Test 14` – Envoi d'un message d'alerte lors d'une famine de carton.

Description

Comportement lorsqu'un évènement est reçu comme quoi il y a une famine de carton.

Résultat attendu

Un message est envoyé, précisant qu'une famine de carton est survenu.

Module Réseau / émission – `Test 15` – Envoie d'un message lors d'une panne imprimante

Description

Comportement lorsqu'un évènement est reçu comme quoi une imprimante est en panne.

Résultat attendu

Un message est envoyé, précisant qu'une imprimante est en panne.

Module Évènements – `Test 16` – DEL / Arrêt d'urgence

Description

Comportement lorsqu'un message est reçu comme quoi un arrêt d'urgence est demandé.

Résultat attendu

Un message doit être placé dans la boite aux lettres évènements. La couleur de la DEL doit être fixée à rouge.

Module Évènements – `Test 17` – DEL / File d'attente carton pleine

Description

Comportement lorsqu'un message est reçu comme quoi la file d'attente des cartons est pleine.

Résultat attendu

La couleur de la DEL doit être fixée à rouge.

Module Évènements – `Test 18` – DEL / Famine de pièces

Description

Comportement lorsqu'un message est reçu comme quoi il y a famine de pièces

Résultat attendu

La couleur de la DEL doit être fixée à rouge.

Module Évènements – `Test 19` – DEL / Famine de cartons

Description

Comportement lorsqu'un message est reçu comme quoi il y a famine de cartons

Résultat attendu

La couleur de la DEL doit être fixée à rouge.

Module Évènements – `Test 20` – DEL / Taux de pièce défectueuses atteint

Description

Comportement lorsqu'un message est reçu comme quoi le taux de pièces défectueuses a été atteint.

Résultat attendu

La couleur de la DEL doit être fixée à rouge.

Module Évènements – `Test 21` – DEL / Anomalie sur une imprimante.

Description
Lorsqu'un message est reçu comme quoi une imprimante est en panne.
Résultat attendu
La couleur de la DEL doit être fixée à orange.

Module Évènements – `Test 22` – DEL / Réparation sur une imprimante.

Description
Lorsqu'un message est reçu comme quoi une imprimante est réparée.
Résultat attendu
La couleur de la DEL doit être fixée à vert.

Module Évènements – `Test 23` – DEL / Reprise sur erreur.

Description
Comportement lorsqu'un message est reçu comme quoi la production reprend.
Résultat attendu
La couleur de la DEL doit être fixée à vert.

Module Journalisation – `Test 24` – Envoyer un message de journalisation —

Description
Comportement lorsqu'un message est reçu dans la boite aux lettres des évènements.
Résultat attendu
Un message doit être écrit sur le disque. Le cache du disque doit être vidé, et les données doivent effectivement être présente sur le disque dur.

Module Client Windows – `Test 25` – Connection à un serveur

Description
Entrer l'adresse et le port du serveur, cliquer sur le bouton "connecter".
Résultat attendu
Le client doit se connecter. Le bouton connecter laisse place à un bouton déconnecter. Les différents widgets se dégrisent pour permettre d'agir sur le server.
Remarque : Si une mauvaise addresse, ou une adresse suivant un mauvait format est rentré, le comportement est incertain.

Les tests suivant impliquent que le client soit connecté à un serveur.

Module Client Windows – `Test 26` – Déconnection d'un server

Description

Cliquer sur le bouton "deconnecter".

Résultat attendu

Le client doit se déconnecter. Le bouton déconnecter laisse place à un bouton connecter. Les différents widgets se grisent empechant l'utilisateur d'agir hors connection.

Module Client Windows – `Test 27` – Reception d'un message de log

Description

A l'aide d'un serveur, ou d'un socket serveur, envoyer un message de log.

Résultat attendu

Le message doit s'inscrire dans la liste des messages de log.

Module Client Windows – `Test 28` – Reception d'un message d'acceptation de pièces

Description

A l'aide d'un serveur, ou d'un socket serveur, envoyer un message d'acceptation de pièces.

Résultat attendu

L'indication du nombre de pièces acceptés indiqué doit s'incrémenter de la valeur envoyes dans le méssage.

Module Client Windows – `Test 29` – Reception d'un message de rejet de pièces

Description

A l'aide d'un serveur, ou d'un socket serveur, envoyer un message de rejet de pièces.

Résultat attendu

L'indication du nombre de pièces rejetés doit s'incrémenter de la valeur envoyé dans le méssage, ainsi que la barre de progression indiquant l'atteinte du seuil de pièces rejeté.
Remarque : Le client n'est pas responsable d'avertir l'utilisateur du dépassement de ce seuil. Il s'agit d'un message d'erreur envoyé par le serveur.

Module Client Windows – `Test 30` – Reception d'un avertissement (warning)

.

Description

A l'aide d'un serveur, ou d'un socket serveur, envoyer un avertissement.

Résultat attendu

L'avertissement doit s'inscricre dans la liste des messages de log.

Module Client Windows – `Test 31` – Reception d'un message d'erreur

Description

A l'aide d'un serveur, ou d'un socket serveur, envoyer un message d'erreur.

Résultat attendu

Une boite de dialogue modale doit apparaitre permettant à l'utilisateur de de faire le choix de continuer ou de stopper la production.

Module Client Windows – `Test 32` – Envoi de la commande Launch

Description

CLiquer sur le bouton Launch

Résultat attendu

Un serveur ou une socket serveur doit recevoir un message Launch. Le bouton Launch doit se griser, tandis que le bouton Stop doit se dégriser.

Module Client Windows – `Test 33` – Envoi de la commande Resume

Description

Cliquer sur le bouton Resume

Résultat attendu

Un serveur ou une socket serveur doit recevoir un message Resume. Le bouton Resume doit se griser tandis que le bouton Stop doit se dégriser.

Module Client Windows – `Test 34` – Envoi de la commande Resume

Description

Cliquer sur le bouton Stop

Résultat attendu

Un serveur ou une socket serveur doit recevoir un message Stop. Le bouton Stop doit se griser tandis que les bouton Launch et Resume doivent se dégriser.

Module Client Windows – `Test 35` – Envoi d'une configuration

Description

Remplir les champs de configuration

Résultat attendu

Un serveur ou une socket serveur doit recevoir un message de configuration comportant les bonnes valeurs des champs de configuration. Le seuil d'acceptation de pièces rejetés doit s'actualiser avec la valeur envoyé dans le message de configuration.
Remarque : si aucune valeur n'est rentré, le comportement est incertain.

Module Carton – `Test 36` – Lancement et fin

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
5. Mettre un jeton dans le sémaphore boxHandlingRequest

Résultat attendu

- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- Aucun autre message ne doit avoir été reçu dans cette boîte.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 37` – Cas nominal

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée d'assez de pièces pour remplir quelques carton (moins de 5 cartons, aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
8. Mettre un jeton dans le sémaphore boxHandlingRequest

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 38` – Surcharge de la file d'attente
Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée d'assez de pièces pour remplir quelques carton (plus de 5 cartons, aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Si un événement `ERR_FULL_QUEUE` est reçu dans un temps raisonnable (par ex. 1s après la dernière pièce sur un système non chargé)
   (a) Retirer un carton de la file
   (b) Mettre un jeton dans le sémaphore boxHandlingRequest
   (c) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (d) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (e) Mettre un jeton dans le sémaphore boxHandlingRequest
8. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 39` – Famine de pièce

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée de quelques pièces, mais pas assez pour remplir un carton (aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Si un événement `EVT_ERR_PRODUCT_STARVATION` est reçu dans un temps raisonnable (par ex. 15s après la dernière pièce sur un système non chargé)
   (a) Mettre un jeton dans le sémaphore boxHandlingRequest
   (b) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (c) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (d) Mettre un jeton dans le sémaphore boxHandlingRequest
8. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 40` – Absence de carton

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Simuler l'absence d'un carton
6. Si un événement `EVT_ERR_BOX_STARVATION` est reçu dans un temps raisonnable (par ex. 1s après l'ajout du jeton)
   (a) Simuler la présence d'un carton
   (b) Mettre un jeton dans le sémaphore boxHandlingRequest
   (c) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (d) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (e) Mettre un jeton dans le sémaphore boxHandlingRequest
7. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 41` – Arrêt d'urgence
Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Simuler un arrêt d'urgence
6. Si un événement `EVT_EMERGENCY_STOP` est reçu dans un temps raisonnable (par ex. 100ms après l'arrêt d'urgence)
   (a) Mettre un jeton dans le sémaphore boxHandlingRequest
   (b) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (c) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (d) Mettre un jeton dans le sémaphore boxHandlingRequest
7. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'arrêt d'urgence.
- Le clapet doit être fermé à la réception de l'événement associé à l'arrêt d'urgence.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Impression – `Test 42` – Reception des informations concernant le carton à imprimer
Description

1. Créer les objets nécessaires au fonctionnement de `PrintManager`
2. Mettre un carton dans la boîte aux lettres `BoxesQueue`
3. Lancer la tâche `PrintManager`

Résultat attendu

- Le carton deposé avant est retiré de la boîte aux lettres.
- Les bonnes valeurs sont stockées dans `boxData`.

Module Impression – `Test 43` – Impression d'un carton

Description

1. Créer les objets nécessaires au fonctionnement de `PrintManager`
2. Mettre un carton dans la boîte aux lettres `BoxesQueue`
3. Lancer la tâche `PrintManager`

Résultat attendu

- Le carton deposé avant est retiré de la boîte aux lettres.
- Le carton est imprimé avec l'imprimante `Printer1`.
- Après impression, un message `EVT_BOX_PRINTED` est deposé dans la boîte aux lettres `EventsQueue`

Module Impression – `Test 44` – Une imprimante est en panne

Description

1. Créer les objets nécessaires au fonctionnement de `PrintManager`
2. Mettre un carton dans la boîte aux lettres `BoxesQueue`
3. Mettre l'imprimante `Printer1` hors service
4. Lancer la tâche `PrintManager`

Résultat attendu

- Le carton deposé avant est retiré de la boîte aux lettres.
- On essaye sans succès d'imprimer le carton avec `Printer1`.
- Un message `EVT_ANOMALY_PRINTER1` est deposé dans la boîte aux lettres `EventsQueue`.
- Le carton est delegué à l'imprimante `Printer2`.
- Après impression avec `Printer2`, un message `EVT_BOX_PRINTED` est deposé dans la boîte aux lettres `EventsQueue`

Module Impression – `Test 45` – Les deux imprimantes sont en panne
Description

1. Créer les objets nécessaires au fonctionnement de `PrintManager`
2. Mettre un carton dans la boîte aux lettres `BoxesQueue`
3. Mettre l'imprimante `Printer1` hors service
4. Mettre l'imprimante `Printer2` hors service
5. Lancer la tâche `PrintManager`

Résultat attendu

- Le carton deposé avant est retiré de la boîte aux lettres.
- On essaye sans succès d'imprimer le carton avec `Printer1`.
- Un message `EVT_ANOMALY_PRINTER1` est deposé dans la boîte aux lettres `EventsQueue`.
- Le carton est delegué à `Printer2`.
- On essaye sans succès d'imprimer le carton avec `Printer2`.
- Un message `EVT_ANOMALY_PRINTER2` est deposé dans la boîte aux lettres `EventsQueue`.
- La tâche attend la réparation d'une imprimante en vérifiant en boucle infinie les états des deux imprimantes.

Module Impression – `Test 46` – Fin d'application
Description

1. Créer les objets nécessaires au fonctionnement de `PrintManager`
2. Positionner le booléen `lastMessage` du message à `TRUE`.
3. Mettre le message dans la boîte aux lettres `BoxesQueue`
4. Lancer la tâche `PrintManager`

Résultat attendu

- Le message est retiré de la boîte aux lettres.
- Un message `EVT_APPLICATION_STOP` est deposé dans la boîte aux lettres `EventsQueue`.
- La tâche se suspend.

Module Journalisation – `Test 47` – Envoyer un message de journalisation —
Description
Lancer le système, et attendre que quelques évènements surviennent.
Résultat attendu
Pour chaque évènement, un message doit être écrit dans le fichier de journalisation, sur le disque.

Module Tache Mère – `Test 48` – Lancement

Description

1. Créer les objets partagés (sémaphores, boîtes aux lettres, socket)
2. Lancer les taches *filles*

Résultat attendu

- L'affichage de la liste des IPC du système montre que les deux sémaphores et trois boîtes aux lettres ont été créés
- Il est possible d'ouvrir une connexion au socket créé
- L'affichage de la liste des taches du système montre que les cinq taches filles ont été créées et sont en attente.

Module Tache Mère – `Test 49` – Terminaison

Description

1. Attendre le sémaphore de synchronisation de fin (*endSync*)
2. Détruire les tâches
3. Détruire les objets partagés
4. Quitter l'application

Résultat attendu

- La tache attend un jeton dans le sémaphore *endSync*
- Les cinq taches *filles* ont été supprimées
- La connexion au socket a été interrompue par le serveur
- L'affichage de la liste des IPC montre que les deux sémaphores et trois boîtes aux lettres n'existent pas
- La tache *masterTask* a retourné un code de fin

Deuxième partie

Tests d'intégration

1 Général

Module Général – `Test 1` – **Ordre de lancement**

Description

Un ordre de lancement est envoyé au système sans qu'aucune configuration n'ai précédemment été envoyée.

Résultat attendu

L'ordre doit être ignoré et le système rester dans un état d'attente.

Module Général – `Test 2` – **Interface opérateur**

Description

L'application Windows est lancée

Résultat attendu

L'interface de l'application doit présenter :

- L'heure
- Le numéro de carton en traitement
- Le taux de pièce défectueuse

Module Général – `Test 3` – **Erreur**

Description

Une erreur est lancée

Résultat attendu

Le clapet se ferme immédiatement et un dialogue est entamé avec l'opérateur

Module Général – `Test 4` – **Famine de pièces**

Description

Le système est en marche mais plus aucune pièce n'est disponible

Résultat attendu

Une erreur est lancée et une entrée dans le journal est ajoutée.

Module Général – `Test 5` – **Connexion**

Description

L'application Windows vient de démarrer

Résultat attendu

La connexion avec le serveur VxWorks est établie. En cas d'erreur, un message est affiché sur le client Windows est l'opérateur est en charge de trouver le problème.

2 Rebut

Module Rebut – `Test 6` – Visualiser le taux de pièces acceptées

Description

Lancer le programme, et envoyer une configuration depuis le client Windows, envoyer un ordre de lancement

Résultat attendu

À chaque carton traité, le taux de pièces acceptées doit être mis à jour dans l'interface de l'application Windows.
TODO : AJOUTER au LOG

Module Rebut – `Test 7` – Rebuter une pièce si elle est défectueuse

Description

Lancer le programme, envoyer une configuration depuis le client Windows et envoyer l'ordre de lancement

Résultat attendu

Lorsqu'une pièce défectueuse est détectée le clapet de rebut doit s'ouvrir, rejeter la pièce et ajouter une entrée dans le log
TODO : AJOUTER au LOG

Module Rebut – `Test 8` – Rebuter une pièce si elle est défectueuse

Description

Lancer le programme, envoyer une configuration depuis le client Windows et envoyer l'ordre de lancement

Résultat attendu

Lorsqu'une pièce correcte est détectée le clapet de rebut doit rester fermé, accepter la pièce et ajouter une entrée dans le log
TODO : AJOUTER au LOG

Module Rebut – `Test 9` – Seuil maximal de pièce défectueuse dépassé

Description

Lancer le programme, et envoyer une configuration depuis le client Windows, envoyer un ordre de lancement et attendre que le seuil soit dépassé.

Résultat attendu

Le clapet doit être fermée et une erreur lancée.

3 Impression

Module Impression – `Test 10` – **Cas nominal**

Description

Un carton est rempli.

Résultat attendu

L'impression d'une étiquette contenant toutes les information du carton doit être réalisée par l'une des deux imprimante

Module Impression – `Test 11` – **Imprimante en panne**

Description

Le système détecte qu'une imprimante est tombée en panne.

Résultat attendu

une anomalie est générée et le voyant d'anomalie est allumé.

Module Impression – `Test 12` – **Une seule imprimante en panne**

Description

Un carton est rempli, une des deux imprimante est en panne.

Résultat attendu

L'impression d'une étiquette contenant toutes les informations du carton doit être réalisée par l'imprimante fonctionnelle.

Module Impression – `Test 13` – **File pleine**

Description

La file d'attente de carton en attente d'impression est pleine (5 cartons dans la file).

Résultat attendu

Une erreur est générée, et le clapet d'arrivé est fermé.

4 Journalisation

Module Journalisation – `Test 14` – **Cas nominal**

Description

Un évènement générant une entrée dans le fichier de journalisation à été lancé

Résultat attendu

Le journal doit être lisible sur le disque et contenir :

- La date d'ajout au fichier
- L'heure d'ajout au fichier
- Les caractéristiques du lot : Nombre de pièces, le nombre de pièces défectueuses, code défaut, code opération, code opérateur

Module Journalisation – `Test 15` – **Cas nominal**

Description

Toute erreur ou anomalie doit être signalée dans les journaux.

Résultat attendu

Lancer l'application dans un fonctionnement pouvant générer des erreurs et des anomalies
Le journal doit contenir la trace de chacune des erreurs et des anomalies survenues.

5 Arrêt d'urgence

Module Arrêt d'urgence – `Test 16` – **Cas nominal**

Description

Un signal d'arrêt d'urgence est reçu par le système.

Résultat attendu

Le clapet se ferme immédiatement, le système s'arrête , puis une erreur est générée et journalisée. Les informations concernant les cartons et la configuration du lot sont conservée. Enfin un dialogue est entrepris avec l'opérateur. De plus, si cela est possible, les cartons dans le file d'impression sont traités.

Module Arrêt d'urgence – `Test 17` – **Dialogue**

Description

Le dialogue avec l'opérateur a abouti à une reprise.

Résultat attendu

Le système reprend le traitement (sélection des pièces, impression, etc.) normalement comme si aucune erreur n'était survenue.

Module Arrêt d'urgence – `Test 18` – **Dialogue**

Description

Le dialogue avec l'opérateur a abouti à un arrêt.

Résultat attendu

Le système termine l'impression des boîtes dans la fil d'attente (sans en accepter de nouvelles) et s'arrête.

6 Paramétrage

Module Paramétrage – `Test 19` – **Cas nominal**

Description

L'opérateur lance un ordre de paramétrage.

Résultat attendu

L'ordre de paramétrage est pris en compte et une entrée est ajoutée au journal.

Module Paramétrage – `Test 20` – Cas nominal

Description

L'opérateur lance un ordre de paramétrage en précisant que ce lot est le dernier.

Résultat attendu

L'ordre de paramétrage est pris en compte et une entrée est ajoutée au journal. Une fois le lot traité, le système s'arrête et une entrée est ajoutée dans le journal.

Module Paramétrage – `Test 21` – Cas nominal

Description

Un ordre de paramétrage valide à été envoyé. Un ordre de lancement est lancé

Résultat attendu

Le système démarre avec les informations fournies lors de la dernière configuration.

Module Carton – `Test 22` – Lancement et fin

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
5. Mettre un jeton dans le sémaphore boxHandlingRequest

Résultat attendu

- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- Aucun autre message ne doit avoir été reçu dans cette boîte.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 23` – Cas nominal
Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée d'assez de pièces pour remplir quelques carton (moins de 5 cartons, aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
8. Mettre un jeton dans le sémaphore boxHandlingRequest

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 24` – Surcharge de la file d'attente
Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée d'assez de pièces pour remplir quelques carton (plus de 5 cartons, aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Si un événement `ERR_FULL_QUEUE` est reçu dans un temps raisonnable (par ex. 1s après la dernière pièce sur un système non chargé)
   (a) Retirer un carton de la file
   (b) Mettre un jeton dans le sémaphore boxHandlingRequest
   (c) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (d) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (e) Mettre un jeton dans le sémaphore boxHandlingRequest
8. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 25` – Famine de pièce

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Mettre un jeton dans le sémaphore boxHandlingRequest
6. Simuler l'arrivée de quelques pièces, mais pas assez pour remplir un carton (aucune pièce défectueuse), espacées les unes des autres par 10ms.
7. Si un événement `EVT_ERR_PRODUCT_STARVATION` est reçu dans un temps raisonnable (par ex. 15s après la dernière pièce sur un système non chargé)
   (a) Mettre un jeton dans le sémaphore boxHandlingRequest
   (b) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (c) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (d) Mettre un jeton dans le sémaphore boxHandlingRequest
8. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 26` – Absence de carton

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Simuler l'absence d'un carton
6. Si un événement `EVT_ERR_BOX_STARVATION` est reçu dans un temps raisonnable (par ex. 1s après l'ajout du jeton)
   (a) Simuler la présence d'un carton
   (b) Mettre un jeton dans le sémaphore boxHandlingRequest
   (c) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (d) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (e) Mettre un jeton dans le sémaphore boxHandlingRequest
7. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'erreur.
- Le clapet doit être fermé à la réception de l'événement associé à l'erreur.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Carton – `Test 27` – Arrêt d'urgence

Description

1. Créer les objets nécessaires au fonctionnement de `BoxManager`
2. Initialiser la mémoire `settings`
3. Lancer la tâche `BoxManager`
4. Configurer l'application (donner des valeurs correctes à la mémoire `settings`)
5. Simuler un arrêt d'urgence
6. Si un événement `EVT_EMERGENCY_STOP` est reçu dans un temps raisonnable (par ex. 100ms après l'arrêt d'urgence)
   (a) Mettre un jeton dans le sémaphore boxHandlingRequest
   (b) Poursuivre l'alimentation en pièces pour plusieurs cartons
   (c) Positionner le booléen `ApplicationEndRequest` des paramètres à `TRUE`.
   (d) Mettre un jeton dans le sémaphore boxHandlingRequest
7. Sinon, abandonner (échec du test)

Résultat attendu

- Des cartons (pleins) doivent avoir été reçus dans la boîte aux lettres modélisant la file de cartons.
- Des cartons (pleins) doivent également avoir été reçus après la reprise suivant l'arrêt d'urgence.
- Le clapet doit être fermé à la réception de l'événement associé à l'arrêt d'urgence.
- Le nombre cumulé de pièces contenues dans ces cartons doit correspondre au nombre de pièces correctes générées, moins celles restantes dans le dernier carton entamé.
- Le nombre cumulté de pièces defectueuses associées à ces cartons doit correspondre au nombre de pièces défectueueses générées, moins celles générées pour le dernier carton entamé.
- Le dernier message doit être reçu dans la boîte aux lettres modélisant la file de cartons. Il devrait être présent dans cette boîte après un temps raisonnable après la demande de fin, par ex. 5s sur un système non chargé.
- Le clapet doit être fermé à la réception de ce message.
- L'évènement `EVT_END_FILLING` doit avoir été déclenché.

Module Réseau / réception – `Test 28` – Recevoir une configuration depuis le client

Description

Lancer le programme, et envoyer une configuration depuis le client Windows.

Résultat attendu

La configuration doit être reçue.

Module Réseau / réception – `Test 29` – Recevoir un ordre de lancement ——
Description
Lancer le programme, envoyer une configuration, puis envoyer un ordre de lancement depuis le client Windows.
Résultat attendu
La configuration doit être reçu, et le système industriel doit se mettre en fonctionnement, selon la configuration reçue.

Module Réseau / réception – `Test 30` – Dire à l'application de s'arrêter –
Description
Lancer le programme, envoyer une configuration, envoyer un ordre de lancement, puis patienter jusqu'à la fin, et enfin, envoyer un ordre d'arrêt, juste avant le dernier carton.
Résultat attendu
Le programme doit se fermer à la fin du dernier carton.

Module Réseau / réception – `Test 31` – Tester la reprise sur erreur ———
Description
Lancer le programme, envoyer une configuration, envoyer un ordre de lancement, et appuyer sur le bouton d'arrêt d'urgence. Envoyer ensuite une commande de reprise depuis le client Windows.
Résultat attendu
Le système industriel doit se remettre à fonctionner.

Module Réseau / réception – `Test 32` – Message de reprise sur erreur alors que le système n'est pas arrêté
Description
Lancer le programme, envoyer une configuration, envoyer un ordre de lancement. Envoyer ensuite un ordre de reprise sur erreur alors que le programme n'est pas en erreur.
Résultat attendu
Le programme doit ignorer la commande.

Module Réseau / réception – `Test 33` – Envoyer une chaine non valide ——
Description
Lancer le programme, puis envoyer une chaine non valide par le réseau, à l'aider d'un utilitaire de type `netcat`.
Résultat attendu
Le système industriel doit ignorer la commande.

Module Réseau / réception – `Test 34` – Envoyer un ordre de configuration avec des valeur non valides

Description

Lancer le programme, envoyer une configuration non valide depuis un utilitaire de type `netcat`.

Résultat attendu

Le système industriel doit ignorer la commande.

Module Réseau / émission – `Test 35` – Envoyer un message d'acceptation de pièce

Description

Lancer le programme, envoyer une configuration valide, et lancer le système.

Résultat attendu

Pour chaque pièce sensée être acceptée par le système, un message doit être envoyé par le système. Le message doit respecter le protocole.

Module Réseau / émission – `Test 36` – Envoyer un message si trop de pièces sont défectueuses

Description

Lancer le système, et attendre qu'il y ait trop de pièce défectueuse.

Résultat attendu

Quand trop de pièces sont défectueuses, le système doit envoyer un message. Ce message doit respecter le protocole.

Module Réseau / émission – `Test 37` – Envoyer un message si la file d'attente pour l'impression est pleine

Description

Lancer le système, et attendre qu'il y ait trop de cartons dans la file d'attente.

Résultat attendu

Quand trop de cartons sont dans le file d'attente, le système doit envoyer un message. Ce message doit respecter le protocole.

Module Réseau / émission – `Test 38` – Envoyer un message lors d'une famine de pièce

Description

Lancer le système, et attendre une famine de pièce (pas de pièce qui sort depuis n secondes.

Résultat attendu

Un message doit être envoyé, précisant qu'il y a famine. Ce message doit respecter le protocole.

Module Réseau / émission – `Test 39` – **Envoyer un message lors d'une famine de cartons**

Description

Lancer le système, et attendre une famine de cartons (pas de cartons alors qu'un carton est demandé).

Résultat attendu

Un message doit être envoyé, précisant qu'il y a famine. Ce message doit respecter le protocole.

Module Réseau / émission – `Test 40` – **Envoyer un message lorsqu'une imprimante est en panne**

Description

Lancer le système, et attendre qu'une imprimante tombe en panne.

Résultat attendu

La première fois qu'une imprimante est en panne et doit imprimer, un message doit être envoyé, précisant quelle imprimante est en panne. Ce message doit être conforme au protocole.

Module Réseau / émission – `Test 41` – **Envoyer un message de journalisation**

Description

Lancer le système, et attendre que quelques évènements surviennent.

Résultat attendu

Pour chaque évènement, un message doit être envoyé. Ce message doit respecter le protocole.

Module Évènements – `Test 42` – **DEL / Arrêt d'urgence**

Description

Lancer le système, provoquer un arrêt d'urgence

Résultat attendu

La DEL doit devenir rouge.

Module Évènements – `Test 43` – **DEL / File d'attente carton pleine**

Description

Lancer le système, attendre que la file d'attente soit pleine.

Résultat attendu

La DEL doit devenir rouge.

Module Évènements – `Test 44` – **DEL / Famine de pièces**

Description

Lancer le système, attendre une famine de pièce.

Résultat attendu

La DEL doit devenir rouge.

Module Évènements – `Test 45` – DEL / Famine de carton

Description
Lancer le système, attendre une famine de carton.
Résultat attendu
La DEL doit devenir rouge.

Module Évènements – `Test 46` – DEL / Taux de pièce défectueuses atteint —

Description
Lancer le système, attendre attendre que le taux de pièces défectueuse soit atteint.
Résultat attendu
La DEL doit devenir rouge.

Module Évènements – `Test 47` – DEL / Anomalie sur une imprimante.

Description
Lancer le système, attendre qu'une imprimante tombe en panne.
Résultat attendu
La DEL doit devenir orange.

Module Évènements – `Test 48` – DEL / Situation normale. ————————

Description
Lancer le système, attendre une erreur, puis rétablir la situation.
Résultat attendu
La DEL doit devenir rouge ou orange selon la gravité de l'erreur, puis redevenir verte quand le système est réparé.

Module Évènements – `Test 49` – Ajout d'une pièce acceptée ————————

Description
Lancer le système, attendre qu'une pièce soit ajouté, et accepté par le système.
Résultat attendu
Le compteur de pièces dans le carton doit être incrémenté.

Module Évènements – `Test 50` – Ajout d'une pièce refusé ————————

Description
Lancer le système, attendre qu'une pièce soit ajouté, et rejeté par le système.
Résultat attendu
Le compteur de pièces défectueuses doit être incrémenté.

Module Journalisation – `Test 51` – Envoyer un message de journalisation — Description

Lancer le système, et attendre que quelques évènements surviennent.

Résultat attendu

Pour chaque évènement, un message doit être écrit dans le fichier de journalisation, sur le disque.