Modalités de l'épreuve orale

d'informatique pour la filière MPI

CONCOURS 2023

Ponts ParisTech, ISAE-SUPAERO, ENSTA Paris, TELECOM Paris, MINES Paris,

MINES Saint Étienne, MINES Nancy, IMT Atlantique, ENSAE Paris, CHIMIE ParisTech - PSL

Modalités de l'épreuve orale d'informatique
Filière MPI - CCMP 2023

1 Généralités sur l'épreuve

L'épreuve orale d'informatique du concours évalue les compétences en informatique et peut porter sur tous les aspects des programmes étudiés en MP2I et en MPI. L'épreuve s'appuie principalement sur le programme d'informatique, mais elle peut aussi mobiliser des connaissances apprises dans les autres matières. L'épreuve est toujours une épreuve sur machine, chaque oral faisant intervenir de la programmation dans un ou plusieurs des langages étudiés en MPI (C, OCaml et SQL).

L'épreuve est notée en utilisant le rendu du candidat, mais aussi les interactions entre celui-ci et le jury. Le rendu est composé d'un compte-rendu papier qui doit être complété au fil de l'oral ainsi que du code écrit sur la machine. La clarté des programmes écrits est prise en compte.

On rappelle que si l'interaction avec le jury est une part importante de l'oral, le jury est là pour évaluer la performance des candidats et non les aider dans les exercices. Il peut arriver qu'un jury débloque un candidat en difficulté et ces aides peuvent être prises en compte ou non au moment de l'évaluation. Par exemple, la reformulation d'une question ou une documentation mal écrite n'est pas prise en compte tandis qu'un candidat bloqué, car ignorant un point de cours, peut se voir pénalisé.

2 Format d'un exercice

Les 3h30 de l'oral peuvent être reparties en un ou plusieurs exercices, chaque exercice étant composé d'une ou plusieurs questions autour d'un même thème avec en général une progression de difficulté. Plusieurs exercices peuvent partager un même thème et l'attaquer sous des angles différents (par exemple une approche différente ou des langages différents). Le passage d'un exercice à l'autre est entièrement à la discrétion du jury selon des critères d'avancement du candidat et de durée prévue pour l'exercice. Les candidats doivent ainsi s'attendre à devoir terminer abruptement un exercice, car le temps imparti pour l'exercice est fini.

Le format de chaque exercice est très libre. Le jury peut fournir toutes les questions d'un exercice dès le début de celui-ci sous la forme d'un document papier ou sur l'ordinateur, mais il peut aussi proposer des exercices très "ouverts" où le candidat est face à un problème général. L'exercice se construit dans un dialogue avec le jury, les candidats pouvant faire des propositions sur la manière de procéder que le jury peut accepter ou non — que la proposition soit pertinente ou non — selon ce qu'il veut tester. Enfin, le jury peut aussi proposer des versions intermédiaires à ce qui est décrit plus haut : par exemple commencer par une question ouverte, puis distribuer un bloc de questions puis de nouveau une question ouverte, etc.

3 Fichiers disponibles pour un exercice

Pendant l'oral, le jury met à disposition des candidats un ensemble de fichiers soit directement sur leur ordinateur soit par l'intermédiaire d'une interface web. Des fichiers peuvent être ajoutés à tout moment : avant le début de l'oral, au début d'un exercice ou même pendant l'exercice.

Si le jury rajoute des fichiers pendant l'examen sur l'ordinateur des candidats, il prendra soin de ne pas écraser des fichiers et de prévenir les candidats de cet ajout. Les fichiers mis sur les machines des candidats peuvent comprendre :

- des *fichiers sources* en C, en OCaml ou en SQL ;
- des *fichiers de données* comme des images, des graphes, des textes, etc. Ces données peuvent être dans des formats standards (comme du CSV) ou non. Il n'est pas attendu de connaître ces formats, il sera toujours fourni soit du code qui lit les données, soit des indications précises sur le format qui permettront de lire facilement les données avec les primitives de base du langage dont la documentation sera fournie ;
- des *scripts* bash ou des Makefile simplifiant la compilation ou faisant des tests sur les programmes. Il n'est pas attendu de savoir lire ou écrire ces scripts. Si un candidat doit lancer un script sans l'aide du jury, il aura à disposition une documentation décrivant la commande à entrer dans le *shell* ;
- des documents décrivant soit l'objectif d'un exercice, soit des questions, soit une explication des fichiers fournis, soit de la documentation.

4 Compétences évaluées

Cette section présente maintenant plusieurs types de questions auxquels les candidats peuvent être confrontés. Cet inventaire est là pour aider la préparation des candidats en présentant des compétences exigées soit dans le dialogue avec le jury soit dans les questions, mais il ne s'agit en aucun cas d'un inventaire exhaustif des questions possibles. Des questions dont le format ne respecte pas les types suivants pourront être posées aux candidats. Voici différentes grandes catégories de questions qui peuvent être posées aux candidats (voir sujet zéro pour des exemples) :

- *Architecture du code.* Dans ce type de questions, le sujet décrit un problème et le candidat doit décrire comment il décomposerait le problème en fonctions, classes, modules ou fichiers, quelles seraient les structures de données appropriées pour représenter les données manipulées, etc. Pour ce type de questions, il n'est pas attendu de notions avancées de génie logiciel (qui ne sont pas au programme), mais simplement la capacité à décomposer un gros problème en plusieurs sous-problèmes plus simples et indépendants ainsi qu'à choisir les bonnes structures de données.
- *Réflexion algorithmique.* Dans ce type de questions, le sujet décrit un problème et le candidat doit proposer un algorithme pour s'attaquer au problème. En général, on s'intéressera à trouver un algorithme suffisamment efficace qui donne la bonne réponse, mais,

dans certains cas, le jury pourra aussi poser des questions plus ouvertes. Par exemple, on pourra demander aux candidats qu'ils proposent des heuristiques sur un problème NP-complet ou utiliser un algorithme A*.

- *Lecture de programme.* Le jury peut donner aux candidats des fragments de programmes ou des programmes entiers et leur demander d'expliquer ce que font ces programmes. Ces questions peuvent être posées pour permettre aux candidats de se familiariser avec un code qu'ils devront ensuite éditer, compléter ou utiliser comme source d'inspiration pour leur propre production.
- *Programmation défensive, tests, débogage et commentaires.* Dans ces questions, le candidat dispose d'un code (correct ou non) et il doit rajouter soit des commentaires, soit des tests, soit spécifier les cas où le programme peut échouer, soit corriger le programme pour qu'il n'échoue plus, soit écrire des tests pour tester le code. Noter que le programme à éditer ou corriger peut être un programme écrit pendant l'oral et donc le jury invite fortement les candidats à appliquer des principes de la programmation défensive (expliciter les pré- ou post-conditions des fonctions, tester avec assert, etc.).
- *Écriture ou édition de programme.* Pendant l'oral les candidats auront à écrire des programmes qu'ils devront ensuite exécuter. Il est possible que le jury fournisse une base de code à éditer ou compléter, mais il est aussi possible que les candidats doivent écrire un programme sans avoir de base. Dans les deux cas, le langage de programmation dans lequel les candidats doivent écrire peut être imposé par l'exercice.
- *Preuves et calculs.* Bien que l'épreuve soit une épreuve d'informatique pratique, le jury peut poser des questions plus théoriques comme, par exemple, des preuves de correction, des calculs de complexité, des raisonnements logiques.
- *Retours critiques.* À la fin d'un exercice, on pourra demander aux candidats un retour critique sur ce qu'ils ont fait. Par exemple, le jury pourra demander une comparaison des divers algorithmes qui ont été utilisés sur un même problème.

Certaines de ces catégories peuvent ne pas être mobilisées pendant un oral et, à l'inverse, des questions sortant de ces catégories peuvent être posées.

5 Écriture du compte-rendu

Les candidats devront écrire un compte-rendu d'environ une page au fil de l'épreuve orale. Le contenu précis de ce qui est attendu dans le compte-rendu sera précisé soit par écrit dans un sujet soit dans les entretiens avec le jury. Cependant, pour que les candidats puissent mieux se préparer, voici à titre indicatif ce que le jury attend.

Tout d'abord il faudra penser à noter brièvement toutes les questions des examinateurs pour pouvoir relire correctement le compte-rendu. Ensuite, pour les questions qui demandent du code, on précisera le chemin et/ou la portion du fichier écrit ou modifié. On précisera aussi si des tests ont été effectués sur le code. Pour les questions n'attendant pas du code, on cherchera à donner un résumé de ce qui a été expliqué au jury et on pourra utiliser des schémas. Il n'est pas attendu des candidats d'écrire par eux-mêmes une conclusion au compte-rendu, mais le jury peut le demander.

6 Environnement de développement

Les épreuves se déroulent dans des salles équipées d'ordinateurs avec au moins un ordinateur par candidat. Un sujet zéro et une image virtuelle très similaire à l'environnement disponible pendant le concours seront disponibles sur le site du concours.

Il n'est pas attendu des candidats qu'ils soient très familiers de l'environnement du concours, mais un candidat qui n'aurait jamais utilisé un OS similaire à celui fourni ou qui n'aurait utilisé aucun des éditeurs présents dans l'environnement risquerait de perdre du temps pendant l'épreuve. De la même façon, comme les candidats auront à exécuter des commandes, il est recommandé qu'ils sachent naviguer dans le système de fichiers à l'aide des commandes `cd`, `ls` et `pwd`.

Les candidats auront accès, dans l'environnement de développement, à la documentation officielle des langages étudiés, à une petite documentation de l'environnement de développement, ainsi qu'à une sélection d'éditeurs de texte. L'environnement de développement ne comprendra pas tous les éditeurs existants, en particulier les éditeurs trop exotiques, non libres ou non supportés sur l'OS choisi ne seront pas inclus.