Documentation Utilisateur

Valentin Laclautre, Anthony Dard, Damien Trouche, Martin Gangand,
Basel Darwish Jzaerly

Table des matières

1 Description du compilateur

Ce compilateur a pour but de compiler des fichiers terminant en *.deca* respectant la syntaxe décrite dans les spécifications du langage deca. Ce compilateur vient avec de nombreuses options qui seront décrites dans la section suivante. Il peut compiler plusieurs programmes en une fois, en spécifiant tous les programmes en argument. Le programme crée par défaut un fichier *.ass* au même endroit que le fichier *.deca*. Ce fichier correspond au programme assembleur pouvant être lu par la machine virtuelle ima. Il est aussi possible de compiler le programme deca en bytecode java (voir section suivante pour plus de détails), afin de pouvoir l'executer avec la machine virtuelle java.

2 Commandes et options

2.1 Commandes decac

decac [-p |-v] [-n] [-r X] [-d]* [-P] fichiers.deca... |[-b]
decac -java [-d]* [-P] fichier deca... |[-b]

2.2 Option -b

Cette option n'est executée que de la manière suivante : **decac -b**. Elle permet d'afficher le numéro du groupe ainsi que les noms des différents membres de l'équipe ayant développé ce compilateur.

2.3 Option -p

L'option **-p** permet de parser le (ou les) fichier(s) deca donné(s) en paramètre(s). Après cette étape, si aucune erreur n'est survenue, le programme est décompilé et est affiché. Il doit être syntaxiquement correct après la décompilation. Cette option ne peut pas être utilisée avec l'option **-v**.

2.4 Option -v

Cette option permet d'effectuer l'étape de vérification de l'arbre. Rien n'est affiché s'il n'y a pas d'erreur. Cette option ne peut pas être utilisée avec l'option **-p**.

2.5 Option -r X

Cette option permet d'effectuer une compilation générant un code source qui n'utilise que X registres banalisés pour s'exécuter. Attention : X doit être compris entre 4 et 16. Dans le cas contraire, une "UnsupportedOperationException" est levée lors de la compilation. Elle indique le message suivant : "You have to use a number between 4 and 16 after -r". Cette option ne peut pas être utilisée avec l'option -java.

2.6 Option -d

Cette option permet d'obtenir des traces de debug : il s'agit de messages s'affichant sur la sortie standard permettant d'en savoir plus ce qu'il s'est passé au cours de la compilation. On peut répéter plusieurs fois l'option (par exemple : decac -d -d -d), et ce jusqu'à trois fois, pour avoir de plus amples informations.

2.7 Option -n

Cette option permet de compiler un programme deca sans faire les tests ci-après à l'exécution : division entière (et reste de la division entière) par 0, débordement arithmétique sur les flottants (inclut la division flottante par 0.0), absence de return lors de l'exécution d'une méthode, conversion de type impossible, déréférencement de null, débordement mémoire (pile ou tas).

2.8 option -java

Cette option correspond à l'extension BYTE du compilateur decac. Elle permet de compiler le fichier deca passé en paramètre en bytecode java. Les fichiers créés en sortie sont au format ".class". Pour chaque classe implémentée dans le fichier deca passé en paramètre, un fichier .class est créé. Il peut donc y avoir plusieurs fichiers .class pour un seul fichier deca. Le programme principal est compilé avec le même nom que que le fichier deca (sauf que l'extension est .class au lieu de .deca) tandis que les classes sont chacune compilées dans un fichier portant le nom de cette classe. Pour compiler et lancer un fichier *fich.deca*, il faut donc taper la commande suivante (dans le dossier du fichier) : *decac -java fich.deca* puis lancer *java fich*. Vous devez aussi disposer au minimum de la version 1.8 de java pour lancer ce programme.

3 Messages d'erreurs

Lors de la vérification de l'arbre, un grand nombre d'erreurs peut survenir lors de la vérification syntaxique ou contextuelle du programme deca. Cette section est consacrée à la description de toutes ces erreurs, afin que vous puissiez les résoudre plus facilement.

3.1 IncludeFileNotFound

3.1.1 [...] : include file not found

Cette erreur survient lorsque le fichier à importer n'est pas trouvé ou impossible à lire.

3.2 InvalidLValue

3.2.1 left-hand side of assignment is not an lvalue

Cette erreur survient lorsque l'utilisateur tente d'assigner une valeur à une opérande qui n'est pas une lvalue. C'est à dire à laquelle il est impossible d'affecter une valeur.

3.3 CircularInclude

3.3.1 Circular include for file [...]

Cette erreur survient lorsque l'utilisateur fait une inclusion de fichier circulaire. C'est à dire lorsqu'un fichier en inclut un autre qui a déjà été inclut.

3.4 ContextualError

3.4.1 (0.1) The identifier is not declared

Cette erreur survient lorsque l'utilisateur a tenté d'utiliser une variable qui n'a pas été déclarée avant. Celle-ci n'est donc pas initialisée et est inconnue. Pour la résoudre, il faut déclarer cette variable.

3.4.2 (0.1) The field is not declared

Cette erreur survient lorsque l'utilisateur utilise un champ qui n'a pas été déclaré. Le champ n'existe pas, il faut donc le créer.

3.4.3 (0.2) The identifier has an invalid type

Cette erreur survient lorsque l'utilisateur tente d'utiliser un type inconnu. En effet, les types autorisés sont : int, float, booléen, et les classes.

3.4.4 (1.3) Super class must be declared before its children

Cette erreur survient lorsque l'utilisateur déclare une classe qui hérite d'une classe mère, avant même que cette classe mère n'ait été déclarée. Il faut d'abord déclarer la classe mère, puis déclarer la classe fille.

3.4.5 (1.3) The class has already been declared

Cette erreur survient lorsque l'utilisateur déclare une classe alors qu'une classe de même nom existe déjà.

3.4.6 (2.5) Field type cannot be void

Cette erreur survient lorsque l'utilisateur déclare un champ de type void. Les type autorisés sont int, float, boolean, et les classes.

3.4.7 (2.5) Name has already been declared

Cette erreur survient lorsque l'utilisateur déclare un champ avec un nom qui est déjà utilisé.

3.4.8 (2.7) Method has already been declared

Cette erreur survient lorsque l'utilisateur déclare une méthode, alors qu'une méthode de même nom existe déjà. Cela est impossible, même si la signature et le type de retour sont différents.

3.4.9 (2.7) Redefinition of a method must match the signature and return a subtype of the super definition

Cette erreur survient lorsque l'utilisateur redéfinit une méthode mais la signature ne correspond pas à celle de la méthode redéfinie. Cette erreur survient aussi si le type de retour n'est pas un sous-type du type de retour de la méthode héritée.

3.4.10 (2.9) Param cannot have void type

Cette erreur survient lorsque l'utilisateur utilise un paramètre de type void. Par exemple, dans la signature d'une méthode, un paramètre doit être de type int, float, boolean, ou une classe.

3.4.11 (3.17) The identifier is already declared

Cette erreur survient lorsque l'utilisateur déclare une variable qui a déjà été déclarée préalablement. En effet, il est impossible de déclarer deux variables qui ont le même nom, même si leur type est différent.

3.4.12 (3.17) Variable declaration with type void is forbidden

Cette erreur survient lorsque l'utilisateur déclare une variable avec le type void, il est impssible d'effectuer cela. Le type d'une variable doit être int, float, boolean, ou une classe.

3.4.13 (3.24) Return value can't be void

Cette erreur survient lorsque l'utilisateur retourne une valeur de type void. Le type de retour doit être int, float, boolean, ou une classe.

3.4.14 (3.28) Type [...] can't be assigned to [...]

Cette erreur survient lorsque l'utilisateur tente d'affecter une opérande à une variable dont le type est incompatible. Par exemple, il n'est pas possible d'affecter un booléen à un float. Cependant, il est possible d'affecter un int à une variable de type float, dans ce cas la conversion est implicite.

3.4.15 (3.29) Condition must return a boolean

Cette erreur survient lorsque l'utilisateur utilise une condition dont le type de retour n'est pas un booléen.

3.4.16 (3.31) Wrong type for the print function. It should be int, float or string

Cette erreur survient lorque l'utilisateur a tenté d'utiliser une fonction d'affichage (print, println, printx, printlnx), mais qu'il a rentré un mauvais type en paramètre. En l'occurence, les seuls type autorisés dans une fonction d'affichage sont : int ou float.

3.4.17 (3.33) Arithmetic operation only : [...] accept (int, int) as operands type

Cette erreur survient lorsque l'utilisateur tente d'effecter l'opération arithmétique modulo entre deux opérandes qui ne sont pas de type int. En effet l'opération arithmétique modulo n'est autorisée qu'entre 2 entiers.

3.4.18 (3.33) Arithmetic operation : [...] only accept ([int—float], [int—float]) as operands type

Cette erreur survient lorsque l'utilisateur tente d'effectuer une opération arithmétique sur des opérandes qui ne sont pas de de type int ou float. En effet, les opérateurs arithmétiques +, -, *, et / n'acceptent que des entiers et des flottants.

3.4.19 (3.33) Boolean operation : [...] only accept ([int—float], [int—float]) or objects for == and !=, as operands type"

Cette erreur survient lorsque l'utilisateur utilise l'opérateur "==" ou " !=" avec des types différents de int et float, ou qui ne sont pas des objets. En effet, il n'est possible de tester une égalité ou inégalité que sur des entiers, des flottants, ou des objets.

3.4.20 (3.33) Comparison operation : [...] only accept ([int—float], [int—float]) or objects for a comparison, as operands type

Cette erreur survient lorsque l'utilisateur utilise un opérateur de comparaison avec des types différents de int et float, ou qui ne sont pas des objets. En effet, il n'est possible de faire une comparaison que sur des entiers, des flottants, ou des objets.

3.4.21 (3.37) Not operator only accept boolean operand

Cette erreur survient lorsque l'utilisateur utilise l'opérateur unaire "not" sur un type différent de booléen.

3.4.22 (3.37) UnaryMinus operator only accept int or float operand

Cette erreur survient lorsque l'utilisateur utilise l'opérateur unaire "-" sur un type différent de int ou float.

3.4.23 (3.39) Can't cast void type

Cette erreur survient lorsque l'utilisateur utilise un cast vers un type void, ce n'est pas possible.

3.4.24 (3.39) Can't cast [...] to [...]

Cette erreur survient lorsque le cast d'un type vers un autre type n'est pas possible. Par exemple, il n'est pas possible de caster un entier en booléen, alors qu'il est possible de caster un entier en boolean.

3.4.25 (3.40) Left side of instanceof must have class type

Cette erreur survient lorsque le type sur l'opérande à gauche de "intanceof" n'est pas une classe. En effet, il n'est pas possible d'utiliser "instanceof" sur les types int, float, et boolean.

3.4.26 (3.40) Right side of instanceof must be a class identifier

Cette erreur survient lorsque l'opérande à droite de "intanceof" n'est pas une instance de classe. En effet, il n'est pas possible d'utiliser "instanceof" sur les types int, float, et int.

3.4.27 (3.42) New can only be used on class type

Cette erreur survient lorsque l'utilisateur utilise "New" sur un type qui n'est pas une classe. En effet, il faut utiliser ce mot clé pour créer une instance d'une classe.

3.4.28 (3.66) Protected fields must be used only in class body

Cette erreur survient lorsque l'utilisateur utilise un champ dont la visibilité est "Protected" en dehors du corps de la classe. Pour la résoudre, il est possible de créer un getter ou un setter sur ce champ, et ainsi pouvoir y accéder en dehors de la classe. Il est aussi possible de changer la visibilité en "public".

3.4.29 (3.70) Indentifier must be a field

Cette erreur survient lorsque l'utilisateur utilise une varibale qui n'est pas un champ.

3.4.30 (3.70) Invalid field indentifier

Cette erreur survient lorsque le champ utilisé n'est pas trouvé, cela peut être à cause d'une visibilité trop restrictive qui empêche d'accéder à ce champ.

3.4.31 (3.71) Method call must match the number of arguments

Cette erreur survient lorsque l'utilisateur fait appel à une méthode avec un nombre d'arguments différents de celui dans la signature de la méthode. Vérifiez bien la signature de la méthode appelée.

3.4.32 (3.71) Method can only be called on class type

Cette erreur survient lorsque l'utilisateur utilise une méthode de classe, mais sans une instance de cette classe. Pour la résoudre il faut créer une instance de cette classe, puis appeler la méthode avec cette instance.

3.4.33 (3.72) Indentifier must be a method

Cette erreur survient lorsque l'utilisateur utilise un identifier qui ne correspond pas à une méthode.

3.4.34 (3.72) Invalid indentifier

Cette erreur survient lorsque l'utilisateur utilise une méthode mais que cette dernière n'est pas dans l'environnement. Une solution peut être de déclarer la méthode en public.

3.4.35 Field has already been declared

Cette erreur survient lorsque l'utilisateur déclare un champ mais que ce dernier a déjà été déclaré.

3.4.36 Param has already been declared

Cette erreur survient lorsque l'utilisateur déclare un paramètre, alors que ce dernier est déjà déclaré.

3.4.37 The -java option can't be used with ima compiler

Cette erreur survient lorsque l'utilisateur utilise decac avec l'option -java sur un programme qui contient un corps de méthode en assembleur.

3.4.38 The option must be -java in order to use java compiler

Cette erreur survient lorsque l'utilisateur utilise decac sans l'option -java sur un programme qui contient un corps de méthode en java.

3.4.39 This cannot be used in main program

Cette erreur survient lorsque l'utilisateur utilise "This" dans le programme principal. Il faut l'utiliser dans une classe.

3.4.40 Cannot select field on [...]

Cette erreur survient lorsque l'utilisateur utilise un champ sur un type qui n'est pas une classe. Par exemple, les type int, float, et boolean n'ont pas de champ. Pour la résoudre, il faut appeler le champ sur une classe.

3.5 Erreurs de compilation

3.5.1 Error : StackOverflow

Cette erreur survient lorsque la taille de la pile est dépassée.

3.5.2 Error : Overflow

Cette erreur survient lorsque l'utilisateur fait un calcul et que le résultat obtenu est trop grand. C'est à dire qu'il n'est pas codable comme un entier signé positif sur 32 bits.

3.5.3 Error : Input / Output error

Cette erreur survient lorsque la valeur saisie par l'utilisateur suite à un readInt() n'est pas un entier, ou lorsque la valeur saisie suite à un readFloat() n'est pas un flottant.

3.5.4 Error : Division by zero

Cette erreur survient lorsque l'utilisateur effectue une division par zero.

3.5.5 Error : Impossible cast

Cette erreur survient lorsque l'utilisateur fait un cast vers un type qui n'est pas compatible.